2006-12

Development of a three dimensional perfectly matched layer for transient elasto-dynamic analyses

Johnson, Anthony N.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

DISSERTATION

DEVELOPMENT OF A THREE DIMENSIONAL PERFECTLY MATCHED LAYER FOR TRANSIENT ELASTO-DYNAMIC ANALYSES

by

Anthony N. Johnson

December 2006

Co-Dissertation Supervisors: Clyde Scandrett
Steve Baker

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 2006	Dissertation

4. TITLE AND SUBTITLE DEVELOPMENT OF A THREE DIMENSIONAL PERFECTLY MATCHED LAYER FOR TRANSIENT ELASTO-DYNAMIC ANALYSES	5. FUNDING NUMBERS
6. AUTHORS Johnson, Anthony N.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited.	

13. ABSTRACT(*maximum 200 words*)

A time dependent, three dimensional finite element approach to the development of a perfectly matched layer for numerical calculations of surface wave radiation in a half space is presented. The development of this new element required the coupling of a system of linear, second-order, partial differential equations which describe elastic wave propagation into a single weak-form (Galerkin) wave equation, from which the characteristics of a composite finite element matching layer were derived. An important problem of interest, and the motivation for this work, is the optimization of a source for use in a seismo-acoustic sonar for the detection of buried mines. Various source excitations are presented which maximize the energy of the unidirectional Rayleigh wave while suppressing the energy of associated body waves. The hp-adaptive finite element code SAFE-T (Solid Adaptive Finite Element - Transient), a Finite Element Method (FEM) implementation developed by the author utilizing Altair Engineering's Prophlex kernel, is used to perform the numerical computations. Results for radial and vertical wave strengths are given. This work represents an important step forward in the development of tools needed to pursue seismo-acoustic sonar technology for buried mine detection, as well as for the analysis of all three-dimensional, time-dependent elasto-dynamic problems.

14. SUBJECT TERMS Perfectly Matched Layer, Seismo-Acoustic Sonar, Finite Element, Prophlex kernel, Rayleigh Energy, time-dependent elasto-dynamic analyses	15. NUMBER OF PAGES 152
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DEVELOPMENT OF A THREE DIMENSIONAL PERFECTLY MATCHED LAYER FOR TRANSIENT ELASTO-DYNAMIC ANALYSES

Anthony N. Johnson
Major, United States Army
B.S., Fayetteville State University, 1999
M.S., Naval Postgraduate School, 2005

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
December 2006

Author: _____
Anthony N. Johnson

Approved by:

_____	_____
Clyde Scandrett	Steve Baker
Professor of Mathematics	Assoc. Professor of Physics
Dissertation Co-Supervisor	Dissertation Co-Supervisor
_____	_____
Beny Neta	David H. Olwell
Professor of Mathematics	Sr. Lecturer of Sys. Eng.
_____	
Carlos Borges	
Professor of Mathematics	

Approved by: _____
Clyde Scandrett, Chair, Department of Mathematics

Approved by: _____
Julie Filizetti, Associate Provost for Academic Affairs

ABSTRACT

A time dependent, three dimensional finite element approach to the development of a perfectly matched layer for numerical calculations of surface wave radiation in a half space is presented. The development of this new element required the coupling of a system of linear, second-order, partial differential equations which describe elastic wave propagation, together with their related boundary conditions, into a single weak-form (Galerkin) wave equation, from which the characteristics of a composite finite element matching layer were derived. An important problem of interest, and the motivation for this work, is the optimization of a source for use in a seismo-acoustic sonar for the detection of buried mines. Validation of the perfectly matched layer occurs by employing it in a finite element analysis to compute the radiation from a particular transient seismo-acoustic source array and showing that the results agreed with the results of previous field experiments using the same source performed by Naval Postgraduate School students. Various source excitations are presented which maximize the energy of the unidirectional Rayleigh wave while suppressing the energy of associated body waves. Radiation characteristics are analyzed in a linear, isotropic, homogeneous half space with a discrete number of transient seismic sources. The hp-adaptive finite element code SAFE-T (Solid Adaptive Finite Element - Transient), a Finite Element Method (FEM) implementation developed by the author utilizing Altair Engineering's Prophlex kernel, is used to perform the numerical computations. Results for radial and vertical wave strengths are given in terms of their total displacement magnitudes. This work represents an important step forward in the development of tools needed to pursue seismo-acoustic sonar technology for buried mine detection, as well as for the analysis of all three-dimensional, time-dependent elasto-dynamic problems.

THIS PAGE INTENTIONALLY LEFT BLANK

DISCLAIMER

Prophlex is a registered trademark of Altair Engineering Inc.

MatLab is a registered trademark of Mathworks Inc.

Mathematica is a registered trademark of Wolfram Research Inc.

Maple is the registered trademark of the Waterloo Corporation.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to express sincere gratitude to...

God without whom this work would not have been possible.

Darlene, my wonderful wife and love of my life, for her tireless support, personal sacrifice and encouragement. You are a gift from God. I love you.

My wonderful children, Marc Anthony, Benjamin, and Cinnamon; thanks for being the best children a father could ever hope for. I owe you guys big!

My mother, Beverly, for showing me that falling down is not an end, but rather an chance to rise. You told me I could, therefore, I wasn't afraid to try. Thank you.

My Dissertation advisor, Professor Clyde Scandrett for reading and re-reading this work. I especially appreciate your keen insights which made the complex seem so simple. You are truly one of the most brilliant professors I have ever met.

My Dissertation advisor, Professor Steve Baker for introducing me to the world of seismic analysis.

BG James Kays and BG Gary Krahn for giving me this opportunity.

Professor Dave Olwell for introducing me to the faculty of Applied Math.

Professor Beny Neta, Professor Bill Gragg, and Professor Chris Frenzen for taking time to serve on my academic board.

Professor Carlos Borges for keeping it real. Rock on!

My first advisor, Professor Kenneth Jones, for teaching mathematics with such an infectious and fervent enthusiasm that it set me on fire.

Dr. Jim Wilson for showing me what a pastor really looks like.

LTC Michael Smith and LTC Don Outing for being a well of wisdom

Tad Litska and David Durocher of Altair Engineering Inc. for their assistance with Hypermesh and the coding of Prophlex.

To each and every well wisher who inquired on my progress, and patiently listened to me complain about the speed of my research. Thank you.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

The original analytic development of a surface displacement resulting from a vertical surface disturbance is due to Lamb [Ref. 1]. His method intricately synthesized the solution of a point and line pressure pulse varying with nearly impulse time dependence from the periodic solution, and yielded displacements of the surface excited by a transient unit step source. Lamb's work has been extended and explored by many authors; Pekeris(1955) [Ref. 2], Garvin(1960) [Ref. 3], and Graff(1975) [Ref. 4] just to name a few. J. D. Achenbach [Ref. 5] notes in his well known book *Wave Propagation in Elastic Solids*,"In recent years the methods and solutions in Lamb's paper have been cast in a somewhat more elegant form and more detailed computations have been carried out, particularly for loads of arbitrary time dependence". Of particular note are the solutions worked out by Pekeris [Ref. 2]. He explored closed-form solutions for vertical and horizontal surface displacements in response to a transient vertical point surface pressure. However, the solutions were only for unit step and delta functions which are not physically realistic, but rather represented limiting cases. H. M. Mooney [Ref. 6] demonstrated the effects of changing Poisson's ratio in the domain, and showed how to analyze transient arbitrary sources. Mooney presented the results in a general form which permit convenient application to other problems, and most importantly comparisons to numerical methods as well. The

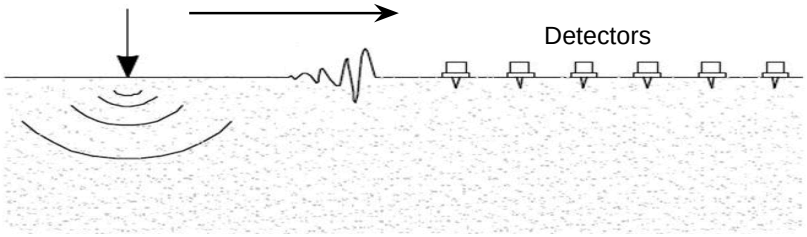


Figure 1. **Surface Wave.**

Finite Element Method (FEM) is a popular tool for the numerical approximation of partial differential equations (PDEs). Because of the availability of powerful and inexpensive computing platforms, the field of computational mechanics has come to rely greatly on FE methods to numerically solve PDEs which arise in the study of various disciplines within the physical sciences [Ref. 7]. Of particular interest to the problem at hand is wave motion in an elastic medium. Seismic wave motions occur in solid media due to elastic forces present in and around solids. A main peculiarity of elastic seismic waves is that in an isotropic medium there can be two or more types of waves that may travel with different velocities and different polarizations. This is due to different elastic states of deformation: mainly shear and compression. Seismic or elastic techniques have shown considerable promise in the reliable detection of various types of buried objects [Ref. 8]. In particular, the study of seismic source array systems to generate surface waves for the detection of land mines in an elastic half-space has provided a unique application for matching mathematical FEM models to results derived through direct experimentation [Ref. 9]. The end goal of the mathematical model is two-fold. First, we want to determine an optimal position, i.e., spacing, of seismic transient sources, and second, we want to accurately time the excitation of seismic sources as to profit from the destructive and constructive interference that results.

B. COMPUTER MODELS

A primary obstacle in formulating an accurate FEM half-space model is addressing how to simulate wave phenomena for the entire unbounded elastic medium. Computer simulations are finite so significant steps must be taken to truncate these systems that attempt to model infinite or semi-infinite domains. Computers are finite machines with limited resources i.e., memory, hard drive space, cpu speed etc. Modeling an infinite domain on a computer system is like attempting to model the effects

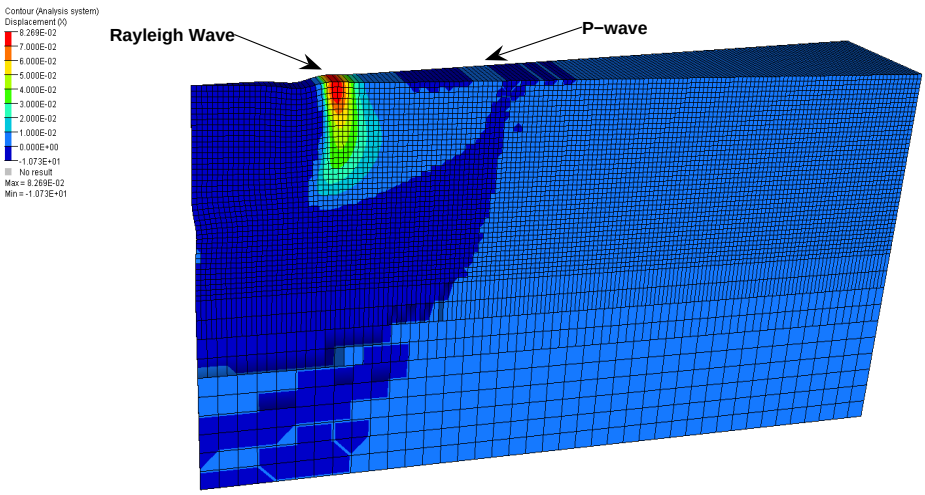


Figure 2. **Axisymmetric wedge of a three dimensional half-space. Modeling an infinite domain on a computer system is like attempting to model the effects and properties of the ocean in a bucket. Axisymmetry is a technique used to conserve resources while not diminishing the scope of the problem.**

and properties of the ocean in a bucket. The challenge, then, is to use a finite space to accurately model an infinite one. Several approaches have been used to accomplish this task. Figure 2 is an attempt to model wave propagation by taking advantage of the axisymmetric properties of the problem. Yet, without significant computer resources to model an enormous domain, even this method falls short of accurately modeling an infinite half-space. In their paper,"Absorbing Boundary Conditions For Acoustic And Elastic Wave Equations," Clayton and Engquist(1977) [Ref. 12] developed an absorbing boundary condition for the 2D elastic wave equation using a paraxial approximation method. With an absorbing boundary, the domain need not be as large and a solution moves closer to within reach. Research continues, however, in improving and extending numerical methodologies of terminating elastic and acoustic unbounded domains. The challenge has led many to attempt the use of traditional frequency domain models in efforts to obtain time domain solutions. One clever approach was explored by Bernal and Youssef(1998) [Ref. 13] who improved the use of a hybrid frequency-time domain procedure which iterates between the frequency and time domain. A reference linear system is solved in the frequency domain

while the equations of motion are solved in the time domain with the unbounded medium represented by frequency independent springs and dampers. The frequency dependency of the impedance of these springs and dampers is introduced into the system by means of frequency domain force evaluations at the end of each time step. Basu(2003) [Ref. 14] comments that this method is computationally demanding, and requires careful implementation to ensure stability.

C. THE PERFECTLY MATCHED LAYER

A modern silent boundary condition that is receiving a flurry of attention is the perfectly matched layer method. First introduced by Berenger[Ref. 15] in 1994 for use with electromagnetic waves, and almost immediately applied by Chew and Weeton[Ref. 16] for use with Maxwell's equations, it has been shown to absorb completely incident plane waves without reflection over a broad range of incidence angles and frequencies. This method is growing rapidly and has been used in many fields, ranging from use with eddy-current problems by Kosmanis'(1999) [Ref. 17] to application to electromagnetic media by Cummer's(2003) [Ref. 18] and linearized shallow water equation models by Navon, Neta, and Hussaini (2001) [Ref. 19]. The reason is that the PML can be formulated at a relatively small computational cost. Recently, Basu and Chopra(2003) [Ref. 11] developed the PML concept in terms of time-harmonic elastodynamics by utilizing insights from electromagnetics and presented a novel displacement based finite element (FE) implementation for time-harmonic plane strain and three-dimensional analysis. Later Basu and Chopra(2004) [Ref. 14] extended the idea to a 2D transient, displacement-based, finite element (FE) method implementation for anti-plane and for plane-strain motion by a special choice of a coordinate stretching technique.

The basic idea behind the PML methodology is to surround the computational domain at the infinite boundary with some type of absorbing layer. As you can see

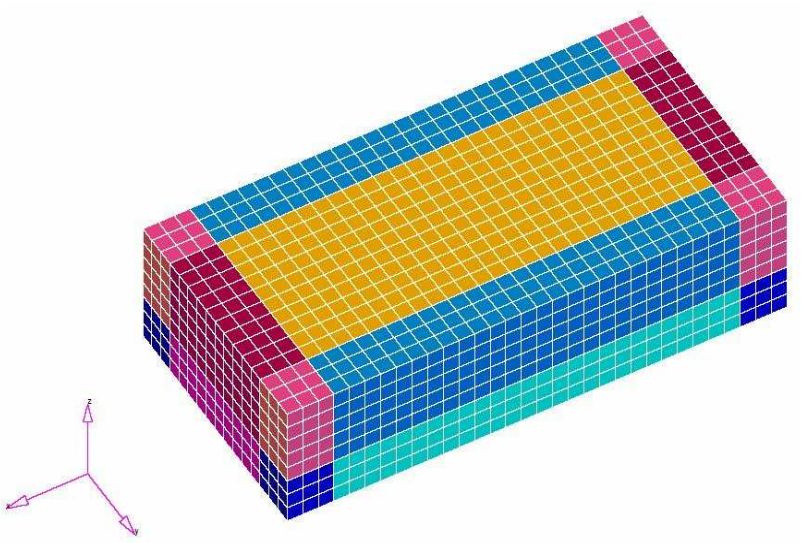


Figure 3. **Perfectly matched layers of a computational domain. The computational domain is in the center surrounded by a boot-like anisotropic sponge which is perfectly matched at the interface between the two domains.**

from Figure 3, the boundary layer is composed of the same kind of elements as the original computational domain. The formulation of the elements is the same for both the computational domain and the absorbing boundary, but the boundary layer has slightly different properties. This boundary layer will be referred to as a perfectly matched layer (PML) which is in substance a perfectly matched media (PMM).

D. DISSERTATION GOAL

It is well established that analytic procedures [Ref. 10] and their software implementations incorporating surface wave interactions are currently formulated in the frequency domain for three dimensions and for up to two dimensions in the time domain [Ref. 11]. This dissertation presents a three dimensional time dependent finite element approach to the development of a perfectly matched layer for numerical calculations of surface wave radiation in a half space. Various source excitations

will be examined to maximize the energy of the unidirectional Rayleigh wave while suppressing the energy of associated body waves. The radiation characteristics are analyzed in a linear, isotropic, homogeneous half-space with a discrete number of transient seismic sources. The mathematical formulation of the problem consists of the coupling of a system of linear, second order, partial differential equations and related boundary conditions into one single wave equation from which a composite elastic element is derived. A time dependent 3D perfectly matched layer (PML) is developed to handle the semi-infinite half space. Outgoing waves are completely absorbed in the PML with minimal reflections from all angles of incidence. The Finite Element Method (FEM) using the hp-adaptive finite element kernel, ProPHLEX, is used to perform the numerical computations.

The following is a brief summary of the remainder of this dissertation. In Chapter II, the major steps involved with developing both analytic and numerical solutions to elastic wave motion problems are outlined. Beginning with the derivation of the elastic partial differential equations used to model wave motion, methods of solving the elastic PDE will focus primarily on solutions which allow the forcing function or source to be arbitrary. A brief introduction to the finite element method will be presented where boundary and initial conditions are constrained to produce a well-posed and useful mathematical model. Chapter II ends with a discussion of the time marching scheme and the mesh conditions employed to make the model dynamic. Chapter III focuses primarily on the complex problem of truncating the computational domain to only allow minimal reflections from the boundary. A terse review of the methodology involved in the implementation of PMLs in the frequency domain is given followed by a more robust three dimensional transformation of the vector quantities to the time domain. Considerable time is spent building the weak or Galerkin forms of the equations. This is necessary in order to implement the mathematical model into the finite element code. Chapter III concludes with SAFE-T's stunning results which showcase the effectiveness of the transient PML and some

sensitivity analysis. Chapter IV is focussed on the specific application of the end-fire array. SAFE-T is used to analyze characteristics of an end-fire array to determine the optimum space and timing that results in the greatest magnitude of the Rayleigh wave while suppressing other surface and non-surface waves. Finding and conclusions are presented in Chapter V along with a discussion about possible future contributions based on the findings of this investigation.

THIS PAGE INTENTIONALLY LEFT BLANK

II. WAVE MOTIONS IN ELASTIC MEDIA

A. INTRODUCTION

This chapter presents the results of a newly developed time dependent finite element (FE) computer code capable of solving 3D vector-valued elastic partial differential equations. A review of techniques used to find analytic solutions is discussed for the surface response to an instantaneous transient point load located one meter away from the source. Finding analytic closed-form solutions for arbitrary surface traction are rare. The analysis of the response to a point or line load acting on a previously tranquil half-space is more common. Care is taken in this chapter to include some of the difficulties involved in forming point or line loaded elastic partial differential equation models while at the same time highlighting the various mathematical tools available to overcome those difficulties. Integral transformations, most useful when an appropriate inversion can be found or evaluated, will play a role in simplifying the model and finding the correct solution. One major challenge to the point load problem, however, is that the solution is non-physical. The point source is, in fact, a singularity which propagates along the free surface of the mathematical model. Singularities are extremely difficult, if not impossible, to model numerically because they involve quantities which are not finite. An integral convolution is utilized to achieve a non-singular result by discretely convolving an arbitrary transient source with the non-physical solution to the point load problem. The result is a new analytic solution that is both realistic and useful. The displacements from the discrete convolution will be compared to the response calculated by the FE computer code when excited by the same source. This chapter presents the first known validation of a time dependent elastic 3D finite element code using a discretely convolved analytic solution.

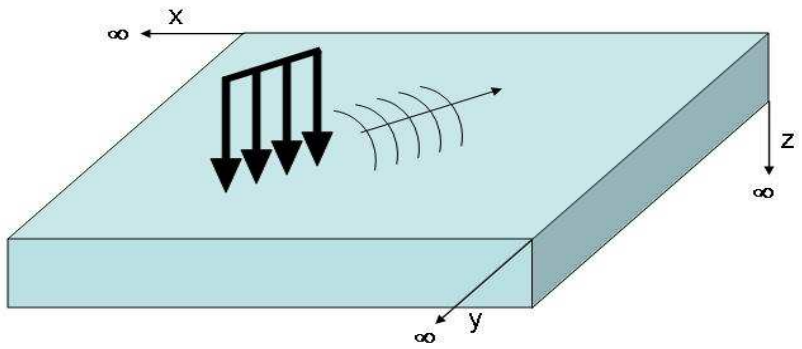


Figure 4. **This graphic depicts the domain of a semi infinite region or half-space. The $x - y$ plane extends from $-\infty$ to $+\infty$ whereas the domain of z extends from the origin to $-\infty$ only.**

B. ANALYTIC FORMULATION

Analytic formulation of an elastic half-space consists of deriving the elastic partial differential equation governing wave motion, determining some method by which to solve the PDE, and finally, finding some way of extending the solution to as general of a forcing function as possible.

1. Derivation of Elastic Partial Differential Equations

In an elastic isotropic medium where the perturbing influences of the surface can be ignored, waves propagate in the form of infinitesimal stress disturbances over a medium [Ref. 20]. An idealized analysis ignores the static body force of gravity, and assumes that the entire medium has a uniform density. Figure 4 is a graphical depiction of this phenomenon. Of interest in an excited elastic material are displacement, stress, strain, and body forces. When body forces are absent, the development of governing equations for the propagation of a linear elastic wave in a solid material begins with the equilibrium equations of motion. In a Cartesian coordinate system, following the convention used by Achenbach[Ref. 5] and Graff[Ref. 4], the equilibrium

equation of motion is expressed as

$$\rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} = \nabla \cdot \boldsymbol{\tau}, \tag{II.1}$$

which translates to

$$\rho \frac{\partial^2 u_x}{\partial t^2} = \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z} \tag{II.2}$$

$$\rho \frac{\partial^2 u_y}{\partial t^2} = \frac{\partial \tau_{yx}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{yz}}{\partial z} \tag{II.3}$$

$$\rho \frac{\partial^2 u_z}{\partial t^2} = \frac{\partial \tau_{zx}}{\partial x} + \frac{\partial \tau_{zy}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} \tag{II.4}$$

and can be written compactly in tensor form as

$$\rho \frac{\partial^2 u_i}{\partial t^2} = \tau_{ij,j}, \tag{II.5}$$

where ρ is the mass density of the material, $u_i = (u_x, u_y, u_z)^T$ is the displacement vector, and τ_{ij} is the stress tensor. Tensor notation is a concise way to express the continuum mechanics of elastic media and will be employed throughout the remainder of this dissertation unless indicated otherwise. In short, a single index denotes a vector, i.e., u_i, and two indices will denote a tensor of order two (a matrix), for example, τ_{ij}. Since Cartesian coordinates are used, all indices denote Cartesian components such that $u_1 = u_x, u_2 = u_y, u_3 = u_z, \tau_{12} = \tau_{xy}$ etc. A summation convention is assumed for *double indices* so that whenever a subscript appears exactly twice in a given term, that subscript will take on the values 1,2,3 successively, and the resulting terms summed. The summed subscripts are simply placeholders or dummy variables since it is immaterial which letter is used. Note, however, in elastic theory there are terms that have more than one pair of dummy indices [Ref. 21]. Partial differentiation with respect to a Cartesian coordinate is denoted by a comma preceding the index [Ref. 22]. Thus

$$u_{i,j} \equiv \frac{\partial u_i}{\partial x_j}, \qquad \phi_{,i} \equiv \frac{\partial \phi}{\partial x_i}, \qquad \psi_{ij,k} \equiv \frac{\partial \psi_{ij}}{\partial x_k} \tag{II.6}$$

From these rules follow that, for example, $u_{i,i}$ is the divergence of the vector u_i i.e., $\left\{\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3}\right\}$ while $u_{,i} = \frac{\partial u}{\partial x_i}$ is the gradient vector $\left\{\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial u}{\partial x_3}\right\}$.

In order to solve II.1, stress must be expressed in terms of strain. Hysteretic analysis of stress and strain shows that a one to one relationship exists between stress and sufficiently small strains in an elastic body [Ref. 23]. Therefore, we can obtain linear relations between stress and strain. Equation II.7 is the constitutive equation representing Hooke's law [Ref. 24] that states stress is proportional to strain

$$\boldsymbol{\tau} = \tau_{ij} = c_{ijkl}\epsilon_{kl} = \lambda\epsilon_{kk}\delta_{ij} + 2\mu\epsilon_{ij}, \tag{II.7}$$

where c_{ijkl} is the fourth order elasticity tensor, whose elements are $(i, j, k, l = x, y, z)$, ϵ_{ij} is the strain tensor, λ and μ are modulus constants, and δ_{ij} is the well known second rank tensor known as the *Kronecker delta*, whose components are defined as

$$\delta_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases} \tag{II.8}$$

We can relate the small-strain tensor to displacements within the limitations of the linearized theory of deformation by

$$\epsilon_{kl} = \frac{1}{2}(u_{k,l} + u_{l,k}). \tag{II.9}$$

Substituting the strain-displacement relations into Hooke's law yields

$$\tau_{ij} = \lambda u_{k,k}\delta_{ij} + \mu(u_{i,j} + u_{j,i}). \tag{II.10}$$

which is

$$\tau_{ij} = \lambda(u_{1,1} + u_{2,2} + u_{3,3}) + \mu(u_{i,j} + u_{j,i}). \tag{II.11}$$

Because of the symmetry of wave propagation in an isotropic, homogeneous media, $u_{i,j} = u_{j,i}$. Therefore,

$$\tau_{ij} = \lambda(u_{1,1} + u_{2,2} + u_{3,3}) + 2\mu u_{i,j}. \tag{II.12}$$

It follows then that by making the substitution into the equation of motion, and using the fact that the elastic stress tensor is symmetric, the elastic wave equation emerges as

$$\rho \frac{\partial^2 u_i}{\partial t^2} = \mu u_{i,jj} + (\lambda + \mu) u_{j,ji}. \tag{II.13}$$

For the purpose of boundary termination the frequency domain wave equation becomes useful and is Fourier transformed to

$$-\rho \omega^2 \bar{u}_i = \mu \bar{u}_{i,jj} + (\lambda + \mu) \bar{u}_{j,ji}. \tag{II.14}$$

so that in a homogeneous, isotropic medium, this equation permits plane-wave solutions of the form

$$\bar{u}_m(x_j, \omega) = A_m e^{i(x_l k_l - \omega t)} \tag{II.15}$$

where A_m represents the amplitude and the polarization of the plane wave, k_l is its wave number in cartesian coordinates, and x_l is the position vector.

2. Method of Solution

As many as three or four different types of waves may exist in an isotropic, homogeneous, elastic media where interfaces and/or free surfaces are present. This contributes to the relative complexity of elastic solid wave problems compared to equivalent problems in acoustics and electromagnetics. Where no body forces exist, II.13 is the displacement equation of motion and represents a system that has the disadvantageous feature of coupling the three displacement components together. To solve this directly would require solving a sixth order partial differential equation. A more convenient and commonly used method for solving the equation would be to express the components of displacement in terms of their potentials and derivatives of potentials [Ref. 5]. Consider a decomposition of the displacement vector of the form

$$u_i = \phi_{,i} + e_{ipq} \psi_{q,p} \tag{II.16}$$

where ϕ is referred to as the scalar potential or *irrotational* portion, curl ψ_q is referred to as the vector potential or *rotational* portion, and e_{ipq} is a common and frequently used rank three *alternating tensor*, whose components are as follows:

$$e_{ipq} = \begin{cases} +1 \text{ if } ipq \text{ represents an even permutation of 123} \\ 0 \qquad\quad \text{if any of the indices are equal} \\ -1 \text{ if } ipq \text{ represents an odd permutation of 123} \end{cases} \tag{II.17}$$

By this decomposition primary variables are

$$u_1 = \phi_{,1} + \psi_{3,2} - \psi_{2,3} \tag{II.18}$$

$$u_2 = \phi_{,2} + \psi_{1,3} - \psi_{3,1} \tag{II.19}$$

$$u_3 = \phi_{,3} + \psi_{2,1} - \psi_{1,2} \tag{II.20}$$

The advantage, of course, of this convention is that it allows us to separate the displacement field into two distinct components, compressional waves and shear waves. We can use this decomposition and solve each potential individually then reconstruct to solve the entire system. We start by substituting II.16 into II.13 to yield

$$\rho(\ddot{\phi}_{,i} + e_{ipq}\ddot{\psi}_{q,p}) = \mu(\phi_{,i} + e_{ipq}\psi_{q,p})_{,jj} + (\lambda + \mu)(\phi_{,j} + e_{jpq}\psi_{q,p})_{,ji}. \tag{II.21}$$

which is further evaluated as (see [Ref. 21])

$$\rho(\ddot{\phi}_{,i} + e_{ipq}\ddot{\psi}_{q,p}) = \mu\phi_{,ijj} + \mu e_{ipq}\psi_{q,pji} + (\lambda + \mu)\phi_{,ijj} + (\lambda + \mu)e_{jpq}\psi_{q,pji}. \tag{II.22}$$

Since $e_{jpq}\psi_{q,pji} = 0$ we obtain

$$\rho(\ddot{\phi}_{,i} + e_{ipq}\ddot{\psi}_{q,p}) = \mu(\phi_{,ijj} + e_{ipq}\psi_{q,pjj}) + (\lambda + \mu)\phi_{,ijj} \tag{II.23}$$

and upon rearranging terms we have

$$((\lambda + 2\mu)\phi_{,jj} - \rho\ddot{\phi})_{,i} + e_{ipq}(\mu\psi_{q,jj} - \rho\ddot{\psi}_q)_{,p} = 0. \tag{II.24}$$

We can see now that II.16 satisfies the equation of motion if

$$(\lambda + 2\mu)\phi_{,jj} - \rho\ddot{\phi} = 0 \tag{II.25}$$

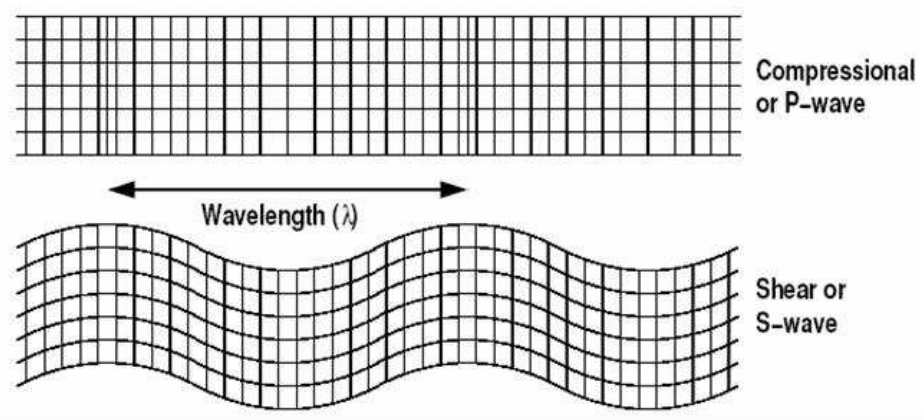


Figure 5. **This graphic is a common depiction used to represent the propagation of P-waves and S-waves in an elastic body. The behavior of P-waves involve compressional or dilatational motion that changes the volume of the material as the wavefront passes while S-wave motion represents a shearing of the material, but does not change the material's volume as the wave progresses.**

and

$$\mu\psi_{q,jj} - \rho\ddot{\psi}_q = 0. \tag{II.26}$$

II.25 and II.26 are the uncoupled wave equations of an elastic media and reveal the total motion as being composed of a curl-free pressure wave (P-wave) traveling with a speed $\left(\frac{\lambda+2\mu}{\rho}\right)^{\frac{1}{2}}$, and a divergence-free shear wave (S-wave) traveling with a speed $\left(\frac{\mu}{\rho}\right)^{\frac{1}{2}}$. By denoting the wave speeds as c_p and c_s respectively, the equations of motion become

$$c_p^2\phi(x_i,t)_{,jj} = \ddot{\phi}(x_i,t) \tag{II.27}$$

and

$$c_s^2\psi(x_i,t)_{q,jj} = \ddot{\psi}(x_i,t)_q. \tag{II.28}$$

Equation II.27 models the behavior of P-waves which involve compressional or dilatational motion that changes the volume of the material as the wavefront passes while equation II.28 models S-wave motion representing a shearing wave that does not change the volume of the material as the wave progresses. Figure 5 is a common depiction used to graphically represent these two motions on an elastic body. If we

take proposed decomposition II.16 and place it back into II.10, the stress components can be written in terms of their displacement potentials as

$$\tau_{ij} = \lambda[\phi_{,kk} + (e_{kpq}\psi_{q,p})_{,k}]\delta_{ij} + \mu\left[\phi_{,ij} + (e_{ipq}\psi_{q,p})_{,j} + \phi_{,ji} + (e_{jpq}\psi_{q,p})_{,i}\right] \quad \text{(II.29)}$$

where symmetry yields

$$\tau_{ij} = \lambda[\phi_{,kk} + (e_{kpq}\psi_{q,p})_{,k}]\delta_{ij} + 2\mu\left[\phi_{,ij} + (e_{ipq}\psi_{q,p})_{,j}\right]. \quad \text{(II.30)}$$

For example, suppose we wanted to know the stress component τ_{11}, then

$$\begin{aligned}\tau_{11} = \lambda[&\phi_{,11} + (e_{1pq}\psi_{q,p})_{,1} + \phi_{,22} + (e_{2pq}\psi_{q,p})_{,2} + \phi_{,33} + (e_{3pq}\psi_{q,p})_{,3}]\delta_{11} \\ &+ 2\mu\left[\phi_{,11} + (e_{1pq}\psi_{q,p})_{,1}\right].\end{aligned}$$

After summing over the p and q we have

$$\begin{aligned}\tau_{11} = \lambda[&\phi_{,11} + (e_{123}\psi_{3,2})_{,1} + (e_{132}\psi_{2,3})_{,1}]\delta_{11} + \lambda[\phi_{,22} + (e_{213}\psi_{3,1})_{,2} + (e_{231}\psi_{1,3})_{,2}]\delta_{11} \\ &+ \lambda[\phi_{,33} + (e_{321}\psi_{1,2})_{,3} + (e_{312}\psi_{2,1})_{,3}]\delta_{11} + 2\mu\left[\phi_{,11} + (e_{123}\psi_{3,2})_{,1} + (e_{132}\psi_{2,3})_{,1}\right].\end{aligned}$$

By taking into account the even and odd permutations of e_{ipq} according to equation II.17 and *Kronecker's delta*, we have

$$\begin{aligned}\tau_{11} &= \lambda[\phi_{,11} + \psi_{3,21} - \psi_{2,31}] + \lambda[\phi_{,22} - \psi_{3,12} + \psi_{1,32}] + \lambda[\phi_{,33} - \psi_{1,23} + \psi_{2,13}] \\ &\quad + 2\mu\left[\phi_{,11} + (\psi_{3,2} - \psi_{2,3})_{,1}\right] \\ \tau_{11} &= \lambda[\phi_{,11} + \phi_{,22} + \phi_{,33} + (\psi_{3,1} - \psi_{3,1})_2 + (\psi_{2,3} - \psi_{2,3})_1 + (\psi_{1,2} - \psi_{1,2})_3] \\ &\quad + 2\mu\left[\phi_{,11} + (\psi_{3,2} - \psi_{2,3})_{,1}\right].\end{aligned}$$

Judicious collection of the ψ terms reveals the cancelation of terms multiplied by λ, simplifying to

$$\tau_{11} = \lambda\phi_{,kk} + 2\mu\left[\phi_{,11} + (\psi_{3,2} - \psi_{2,3})_{,1}\right],$$

and accordingly for the remaining stress components we have,

$$\tau_{22} = \lambda\phi_{,kk} + 2\mu\left[\phi_{,22} - (\psi_{3,1} - \psi_{1,3})_{,2}\right]$$

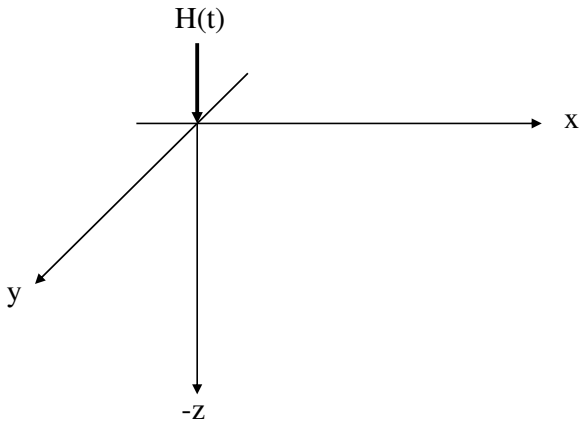


Figure 6. **Point load problem in Cartesian coordinates.**

$$\tau_{33} = \lambda\phi_{,kk} + 2\mu\left[\phi_{,33} + (\psi_{2,1} - \psi_{1,2})_{,3}\right]$$
$$\tau_{12} = \tau_{21} = \mu\left[2\phi_{,12} + (\psi_{3,2} - \psi_{2,3})_{,2} - (\psi_{3,1} - \psi_{1,3})_{,1}\right]$$
$$\tau_{23} = \tau_{32} = \mu\left[2\phi_{,23} - (\psi_{3,1} - \psi_{1,3})_{,3} + (\psi_{2,1} - \psi_{1,2})_{,2}\right]$$
$$\tau_{31} = \tau_{13} = \mu\left[2\phi_{,31} + (\psi_{3,2} - \psi_{2,3})_{,3} + (\psi_{2,1} - \psi_{1,2})_{,1}\right].$$

Thus by solving II.27 and II.28 respectively, we can find not only displacement from II.16, but also the amount of stress at any point as well.

a. The Half-Space

In order to formulate a problem for an elastic half-space ($z \leq 0$), we need boundary conditions at the surface (see Figure 6). Our source will be a point load on the surface $z = 0$, and boundary conditions will be expressed as components of stress. Thus,

$$\tau_{ij} n_j = 0 \tag{II.31}$$

except

$$\tau_{33} = -WH(t)\delta(x)\delta(y) \tag{II.32}$$

where $H(t)$ is the time dependent Heaviside function and W is a force. Together, with the spatial two dimensional delta function $\delta(x)\delta(y)$, τ_{33} specifies a pressure or traction

on the surface. The problem is not completely formulated, however, until we make a statement about the initial conditions. In this case, our medium is undisturbed until some time t where it is excited by a pressure directed downward. This gives initial conditions as

$$\phi(x_i, 0) = \dot{\phi}(x_i, 0) = 0 \tag{II.33}$$

$$\psi_k(x_i, 0) = \dot{\psi}_k(x_i, 0) = 0 \tag{II.34}$$

Taken all together II.33, II.34, along with II.27 and II.28 form a well posed system of PDE's which can be solved individually to find all components of displacement in the elastic media.

b. Integral Transforms

The goal of this section is to obtain integral representations of the potentials. These potentials can then be used in conjunction with the stress representations to find unique solutions to the potentials and thereby determine the half-space surface displacements. Using the one-sided Laplace transformations [Ref. 25], where Br denotes the *Bromwich* inversion path in the complex p-plane,

$$\bar{f}(r, z, \omega) = \int_0^\infty f(r, z, t) e^{-\omega t} dt, \tag{II.35}$$

$$f(r, z, t) = \frac{1}{2\pi i} \int_{Br} \bar{f}(r, z, \omega) e^{\omega t} d\omega, \tag{II.36}$$

all time dependence of equation II.27 and equation II.28 are removed. Thus,

$$\bar{\phi}(x_i, \omega)_{,ii} = \frac{\omega^2}{c_p^2} \bar{\phi}(x_i, \omega) \tag{II.37}$$

and

$$\bar{\psi}(x_i, \omega)_{k,ii} = \frac{\omega^2}{c_s^2} \bar{\psi}(x_i, \omega)_k \tag{II.38}$$

are the integral transform representations for the two potential equations. The next step uses a Double Fourier Transform pair [Ref. 26], given as

$$\tilde{F}(\xi_1, \xi_2) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x_1, x_2) e^{i\xi_1 x_1} e^{i\xi_2 x_2} dx_1 dx_2 \tag{II.39}$$

$$f(x,y) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\tilde{F}(\xi_1,\xi_2)e^{-i\xi_1 x}e^{-i\xi_2 y}d\xi_1 d\xi_2 \tag{II.40}$$

over the entire infinite $x_1(x)$ space and $x_2(y)$ space. Note, also, that

$$e^{i\xi_1 x}e^{i\xi_2 y} = e^{i(\xi_1 x+\xi_2 y)} = e^{i\xi_i\cdot x_i}. \tag{II.41}$$

The $x_3(z)$ dimension remains untransformed in the analysis. This converts the PDEs into ODEs with respect to x_3. For $\bar{\phi}(x_i,\omega)$ and $\bar{\psi}(x_i,\omega)_k$ we have:

$$\bar{\phi}(x_1,x_2,x_3,\omega)_{,ii} = \frac{\omega^2}{c_p^2}\bar{\phi}(x_1,x_2,x_3,\omega) \tag{II.42}$$

$$\bar{\psi}(x_1,x_2,x_3,\omega)_{k,ii} = \frac{\omega^2}{c_s^2}\bar{\psi}(x_1,x_2,x_3,\omega)_k. \tag{II.43}$$

After application of spatial transforms in the $x_1(x)$, $x_2(y)$ coordinates, and integration by parts we derive

$$\tilde{\phi}(\xi_1,\xi_2,x_3,\omega)_{,33} = \left(\frac{\omega^2}{c_p^2} + (\xi_1^2+\xi_2^2)\right)\tilde{\phi}(\xi_1,\xi_2,x_3,\omega) \tag{II.44}$$

$$\tilde{\psi}(\xi_1,\xi_2,x_3,\omega)_{k,33} = \left(\frac{\omega^2}{c_s^2} + (\xi_1^2+\xi_2^2)\right)\tilde{\psi}(\xi_1,\xi_2,x_3,\omega)_k. \tag{II.45}$$

The above equations are now second order ODEs with respect to $x_3(z)$. Solutions of equation II.44 and equation II.45 take the form

$$\tilde{\phi}(\xi_1,\xi_2,x_3,\omega) = \Phi(\xi_1,\xi_2,\omega)e^{\pm\left(\frac{\omega^2}{c_p^2}+(\xi_1^2+\xi_2^2)\right)^{\frac{1}{2}}x_3} \tag{II.46}$$

$$\tilde{\psi}(\xi_1,\xi_2,x_3,\omega)_k = \Psi(\xi_1,\xi_2,\omega)_k e^{\pm\left(\frac{\omega^2}{c_s^2}+(\xi_1^2+\xi_2^2)\right)^{\frac{1}{2}}x_3} \tag{II.47}$$

At this point it is interesting to note that in a homogenous isotropic material elastic waves propagate equally well in all directions. This, of course, is intuitive since the material acts on the speed of the wave in the same way at every point as it propagates through the substance. Therefore, the motion is invariant with respect to ξ_2 if the wave is traveling normal to ξ_1 and vice versa. This means that the complexity of the problem can be reduced significantly by using cylindrical polar coordinates (r,z,θ). Figure 7 shows the mapping of polar (i.e., $\xi_r^2 = \xi_1^2+\xi_2^2,\ z=x_3$) instead of Cartesian

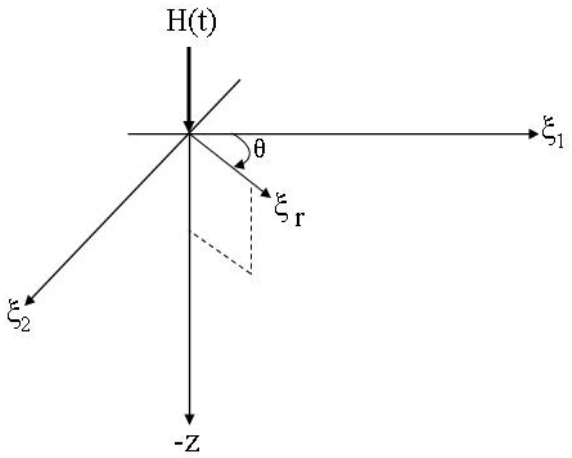


Figure 7. **Point load problem in polar coordinates.**

coordinates where the subscript r in ξ_r denotes radial motion in the $x-y$ plane. For convenience, $\xi = \xi_r$. Primary variable equations II.18, II.19, and II.20 are converted to their polar counterparts as

$$u_r = \phi_{,r} + \psi_{z,\theta} - \psi_{\theta,z} = \frac{\partial \phi}{\partial r} + \frac{\partial \psi_z}{\partial \theta} - \frac{\partial \psi_\theta}{\partial z} \tag{II.48}$$

$$u_\theta = \phi_{,\theta} + \psi_{r,z} - \psi_{z,r} = \frac{1}{r}\frac{\partial \phi}{\partial \theta} + \frac{\partial \psi_r}{\partial z} - \frac{\partial \psi_z}{\partial \theta} \tag{II.49}$$

$$u_z = \phi_{,z} + \psi_{\theta,r} - \psi_{r,\theta} = \frac{\partial \phi}{\partial z} + \frac{1}{r}\frac{\partial (\psi_\theta r)}{\partial r} - \frac{1}{r}\frac{\partial \psi_r}{\partial \theta} \tag{II.50}$$

Since all θ dependence can be in effect removed because wave motion is axially symmetric.; see [Ref. 27] and [Ref. 5], we are essentially left with

$$u_r = \phi_{,r} - \psi_{\theta,z} = \frac{\partial \phi}{\partial r} - \frac{\partial \psi_\theta}{\partial z} \tag{II.51}$$

$$u_z = \phi_{,z} + \psi_{\theta,r} = \frac{\partial \phi}{\partial z} + \frac{1}{r}\frac{\partial (\psi_\theta r)}{\partial r} \tag{II.52}$$

Observe that now we are actually manipulating two uncoupled problems of wave motion. u_r and u_z depend only on the quantities ϕ and ψ_θ which are governed by the independent scalar wave equations II.53 and II.54. We say equation II.54 is scalar because horizontal and vertical displacement equations are computed from only the

ψ_θ component of the vector. Thus,

$$\tilde{\phi}(\xi, z, \omega)_{,zz} = \left(\frac{\omega^2}{c_p^2} + \xi^2\right) \tilde{\phi}(\xi, z, \omega) \tag{II.53}$$

$$\tilde{\psi}(\xi, z, \omega)_{\theta,zz} = \left(\frac{\omega^2}{c_s^2} + \xi^2\right) \tilde{\psi}(\xi, z, \omega)_\theta, \tag{II.54}$$

becomes the two ordinary differential equations of potentials

$$\frac{d^2\tilde{\phi}}{dz^2} = \alpha^2 \tilde{\phi} \tag{II.55}$$

$$\frac{d^2\tilde{\psi}_\theta}{dz^2} = \beta^2 \tilde{\psi}_\theta \tag{II.56}$$

where α and β are $\sqrt{\frac{\omega^2}{c_p^2} + \xi^2}$ and $\sqrt{\frac{\omega^2}{c_s^2} + \xi^2}$ respectively. Solutions to II.55 and II.56 will be of the form

$$\tilde{\phi}(\xi, z, \omega) = \Phi(\xi, \omega) e^{\pm \alpha z} \tag{II.57}$$

$$\tilde{\psi}(\xi, z, \omega) = \Psi(\xi, \omega) e^{\pm \beta z} \tag{II.58}$$

With respect to the new system of cylindrical polar coordinates (r, θ, z), the *axially symmetric* motions in a half-space break down to the following stress equations

$$\tau_{zz} = (\lambda + 2\mu) u_{z,z} + \lambda r^{-1} (r u_r)_{,r} \tag{II.59}$$

$$\tau_{zr} = \mu (u_{r,z} + u_{z,r}) \tag{II.60}$$

where again we have (τ_{zz}, τ_{zr}) as the components of the stress tensor. Next the boundary conditions on the surface have to be converted to cylindrical polar coordinates. Thus,

$$\tau_{zz} = -WH(t) \frac{\delta(r)}{2\pi r} \tag{II.61}$$

and

$$\tau_{rz} = 0 \tag{II.62}$$

are converted to their polar equivalents. Initial conditions properly converted simply become

$$\phi(r, z, 0) = \dot{\phi}(r, z, 0) = 0 \tag{II.63}$$

$$\psi(r,z,0) = \dot{\psi}(r,z,0) = 0 \tag{II.64}$$

Now we use a couple of transform pairs to seal the deal. For time, we keep the one sided *Laplace* transform II.35 and II.36. In place of the double *Fourier* transform (II.39 and II.40) used earlier, the *Hankel* transform is enlisted to do the task because of its axisymmetry and use of Bessel functions. The appropriate definitions are as follows [Ref. 25]

$$L\{f\} = \bar{f}(r,z,\omega) = \int_0^\infty f(r,z,t)e^{-\omega t}dt, \tag{II.65}$$

$$L\{\bar{f}\} = f(r,z,t) = \frac{1}{2\pi i}\int_{Br} \bar{f}(r,z,\omega)e^{\omega t}d\omega, \tag{II.66}$$

$$H_\nu\{\bar{f}\} = \tilde{f}(\xi,z,\omega) = \int_0^\infty \bar{f}(r,z,\omega)J_\nu(\xi r)rdr, \tag{II.67}$$

$$H_\nu\{\tilde{f}\} = \bar{f}(r,z,\omega) = \int_0^\infty \tilde{f}(\xi,z,\omega)J_\nu(\xi r)\xi d\xi \tag{II.68}$$

where equation II.65 and equation II.67 provide the direct transforms and equation II.66 and equation II.68 yield inversions. Br denotes the *Bromwich* inversion path in the complex p-plane, and $J_\nu()$ are Bessel functions of the first kind of order ν. Applying the transforms to the displacements yield

$$\tilde{u}_r^1(\xi,z,\omega) = \int_0^\infty u_r(r,z,\omega)J_1(\xi r)rdr \tag{II.69}$$

$$\tilde{u}_r^1(\xi,z,\omega) = \int_0^\infty \left(\frac{\partial\phi}{\partial r} - \frac{\partial\psi_\theta}{\partial z}\right)J_1(\xi r)rdr \tag{II.70}$$

$$\tilde{u}_r^1(\xi,z,\omega) = \int_0^\infty \left(\frac{\partial\phi}{\partial r}\right)J_1(\xi r)rdr - \int_0^\infty \left(\frac{\partial\psi_\theta}{\partial z}\right)J_1(\xi r)rdr \tag{II.71}$$

$$\tilde{u}_r^1(\xi,z,\omega) = -\xi\int_0^\infty \phi J_0(\xi r)rdr - \frac{d}{dz}\int_0^\infty \psi_\theta J_1(\xi r)rdr \tag{II.72}$$

$$\tilde{u}_r^1(\xi,z,\omega) = -\xi\tilde{\phi}^0(\xi,z,\omega) - \frac{d}{dz}\tilde{\psi}^1(\xi,z,\omega) \tag{II.73}$$

for radial displacement and

$$\tilde{u}_z^0(\xi,z,\omega) = \int_0^\infty u_z(r,z,\omega)J_0(\xi r)rdr \tag{II.74}$$

$$\tilde{u}_z^0(\xi,z,\omega) = \int_0^\infty \left(\frac{\partial\phi}{\partial z} + \frac{\partial\psi_\theta}{\partial r} + \frac{\psi_\theta}{r}\right)J_0(\xi r)rdr \tag{II.75}$$

$$\tilde{u}_z^0(\xi,z,\omega) = \int_0^\infty \left(\frac{\partial \phi}{\partial z}\right) J_0(\xi r) r dr + \int_0^\infty \left(\frac{\partial \psi_\theta}{\partial r} + \frac{\psi_\theta}{r}\right) J_0(\xi r) r dr \tag{II.76}$$

$$\tilde{u}_z^0(\xi,z,\omega) = \frac{d}{dz}\int_0^\infty \phi J_0(\xi r) r dr + \xi \int_0^\infty \psi_\theta J_1(\xi r) r dr \tag{II.77}$$

$$\tilde{u}_z^0(\xi,z,\omega) = \frac{d}{dz}\tilde{\phi}^0(\xi,z,\omega) + \xi\tilde{\psi}^1(\xi,z,\omega) \tag{II.78}$$

for vertical displacement. The necessary stresses, equations II.59 and II.60, are replaced with the equivalent potentials and then similarly transformed

$$\tau_{zz} = (\lambda+2\mu)\left\{\phi_{,zz} + \psi_{,zr} + \frac{1}{r}\psi_{,z}\right\} + \frac{\lambda}{r}\left(\phi_{,r} - \psi_{,z} + r\phi_{,rr} - r\psi_{zr}\right) \tag{II.79}$$

$$\tilde{\tau}_{zz}^0 = (\lambda+2\mu)\tilde{\phi}_{,zz}^0 + 2\mu H_0\{\psi_{,r}\}_{,z} + 2\mu H_0\left\{\frac{1}{r}\psi_{,z}\right\} - \xi^2\lambda\tilde{\phi}^0 \tag{II.80}$$

from equation II.55,$\tilde{\phi}_{,zz}^0 = \alpha^2\tilde{\phi}^0$

$$\tilde{\tau}_{zz}^0 = (\lambda+2\mu)\alpha^2\tilde{\phi}^0 + 2\mu H_0\{\psi_{,r}\}_{,z} + 2\mu H_0\left\{\frac{1}{r}\psi_{,z}\right\} - \xi^2\lambda\tilde{\phi}^0 \tag{II.81}$$

$$\tilde{\tau}_{zz}^0 = (\lambda+2\mu)\left(\frac{\omega^2}{c_p^2} + \xi^2\right)\tilde{\phi}^0 + 2\mu H_0\{\psi_{,r}\}_{,z} + 2\xi\mu\psi_{,z}^1 - \xi^2\lambda\tilde{\phi}^0 \tag{II.82}$$

$$\tilde{\tau}_{zz}^0 = \mu\left(\frac{\omega^2}{c_p^2} + 2\xi^2\right)\tilde{\phi}^0 + 2\xi\mu\frac{\partial\psi^1}{\partial z} \tag{II.83}$$

and

$$\tilde{\tau}_{zr}^1 = -2\xi\mu\frac{\partial\phi^0}{\partial z} - \mu\left(\frac{\omega^2}{c_p^2} + 2\xi^2\right)\tilde{\psi}^1 \tag{II.84}$$

Now we transform the boundary conditions on the surface to become

$$\tilde{\tau}_{zz}^0 = -\frac{W}{2\pi\omega} \tag{II.85}$$

and

$$\tilde{\tau}_{rz}^1 = 0. \tag{II.86}$$

Equations II.57 and II.58 are assumed to be the solutions to $\tilde{\phi}^0$ and $\tilde{\psi}^1$. Therefore, placing the assumed solutions into equations II.83 and II.84 with boundary conditions II.85 and II.86 yields a system of stress equations which can be used to determine Φ and Ψ. Thus,

$$\mu\left(\frac{\omega^2}{c_p^2} + 2\xi^2\right)\Phi(\xi,\omega)e^{-\alpha z} + 2\xi\mu\frac{\partial}{\partial z}\left(\Psi(\xi,\omega)e^{-\beta z}\right) = -\frac{W}{2\pi\omega} \tag{II.87}$$

and

$$-2\xi\mu\frac{\partial}{\partial z}\left(\Phi(\xi,\omega)e^{-\alpha z}\right)-\mu\left(\frac{\omega^2}{c_p^2}+2\xi^2\right)\Psi(\xi,\omega)e^{-\beta z}=0 \tag{II.88}$$

which when evaluated at the surface $z=0$ gives the two equations

$$\mu\left(\frac{\omega^2}{c_p^2}+2\xi^2\right)\Phi-2\xi\mu\left(\frac{\omega^2}{c_p^2}+\xi^2\right)^{\frac{1}{2}}\Psi=-\frac{W}{2\pi\omega} \tag{II.89}$$

and

$$-2\xi\left(\frac{\omega^2}{c_s^2}+\xi^2\right)^{\frac{1}{2}}\Phi+\left(\frac{\omega^2}{c_p^2}+2\xi^2\right)\Psi=0. \tag{II.90}$$

The solutions are

$$\Phi=-\left(\frac{W}{2\pi\mu\omega}\right)\frac{2\xi^2+\frac{\omega^2}{c_p^2}}{\left(2\xi^2+\frac{\omega^2}{c_p^2}\right)^2-4\xi^2\sqrt{\xi^2+\frac{\omega^2}{c_p^2}}\sqrt{\xi^2+\frac{\omega^2}{c_s^2}}} \tag{II.91}$$

and

$$\Psi=-\left(\frac{W}{2\pi\mu\omega}\right)\frac{2\xi\sqrt{\xi^2+\frac{\omega^2}{c_s^2}}}{\left(2\xi^2+\frac{\omega^2}{c_p^2}\right)^2-4\xi^2\sqrt{\xi^2+\frac{\omega^2}{c_p^2}}\sqrt{\xi^2+\frac{\omega^2}{c_s^2}}}. \tag{II.92}$$

This now allows the determination of explicit vertical and radial displacements ([Ref. 5]) by using equations II.73 and II.78, such that

$$\tilde{u}_r^1(\xi,z,\omega)=-\xi\Phi(\xi,\omega)e^{-\alpha z}-\frac{d}{dz}\Psi(\xi,\omega)e^{-\beta z} \tag{II.93}$$

and

$$\tilde{u}_z^0(\xi,z,\omega)=\frac{d}{dz}\Phi(\xi,\omega)e^{-\alpha z}+\xi\Psi(\xi,\omega)e^{-\beta z}. \tag{II.94}$$

c. Analytic Solution

Integral inversions of equations II.93 and II.94 according to integral inversion equation II.68 produce the following equations (see [Ref. 27]) for the Laplace-transformed displacements at the surface, i.e., $z=0$

$$\bar{u}_r(r,0,\omega)=\frac{W}{2\pi\mu\omega}\int_0^\infty\frac{\xi^2\left(2\sqrt{\xi^2+k_p^2}\sqrt{\xi^2+k_s^2}-k_s^2-2\xi^2\right)J_1(\xi r)}{R(\xi,\omega)}d\xi \tag{II.95}$$

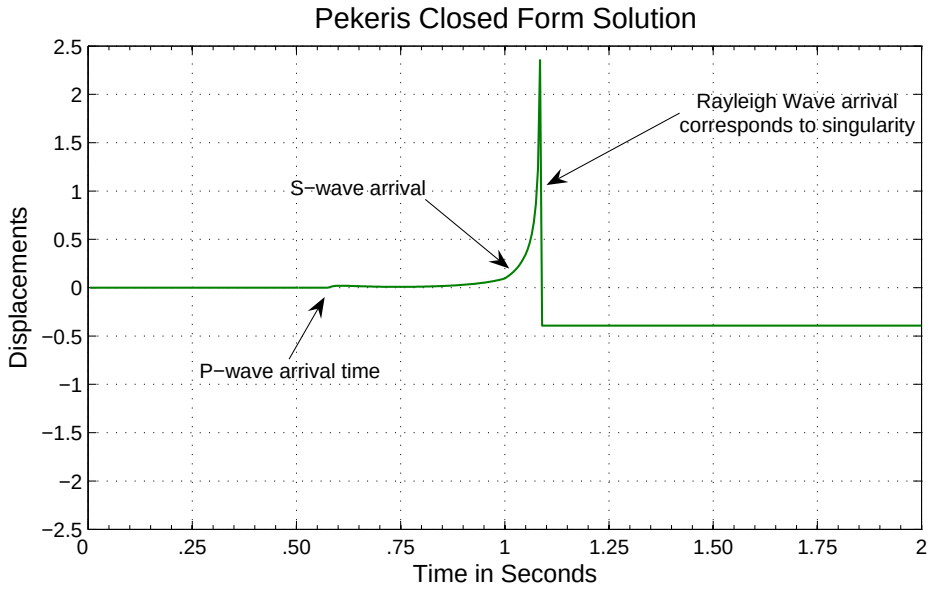


Figure 8. **Vertical surface displacement $u_z(r,z,t)$ due to a concentrated point source directed downward on the surface. Poisson's ratio, ν, is $\frac{1}{4}$ with $\lambda = \mu$. The measurement is located one unit radially from the point of impact. P denotes the arrival time of compressional waves, and S of the shear wave. The Rayleigh wave propagates at the speed of the singularity.**

$$\bar{u}_z(r,0,\omega) = \frac{W}{2\pi\mu\omega}\int_0^\infty \frac{\xi\left(\sqrt{\xi^2+k_p^2}\right)k_s^2 J_0(\xi r)}{R(\xi,\omega)}d\xi \tag{II.96}$$

where $R(\xi,\omega)$ is the Rayleigh function defined as

$$R(\xi,\omega) = (2\xi^2+k_p^2)^2 - 4\xi^2\sqrt{\xi^2+k_p^2}\sqrt{\xi^2+k_s^2} \tag{II.97}$$

and variables k_p, k_s are

$$k_p = \frac{\omega}{c_p} \tag{II.98}$$

and

$$k_s = \frac{\omega}{c_s} \tag{II.99}$$

respectively. The challenge is to perform the transform inversion of the displacement integrals. Many have accomplished this with various techniques. There have been numerical approaches [Ref. 28] as well as closed-form solutions. Most notable of the closed-form solutions would be that of Pekeris [Ref. 2]. Achenbach [Ref. 5] con-

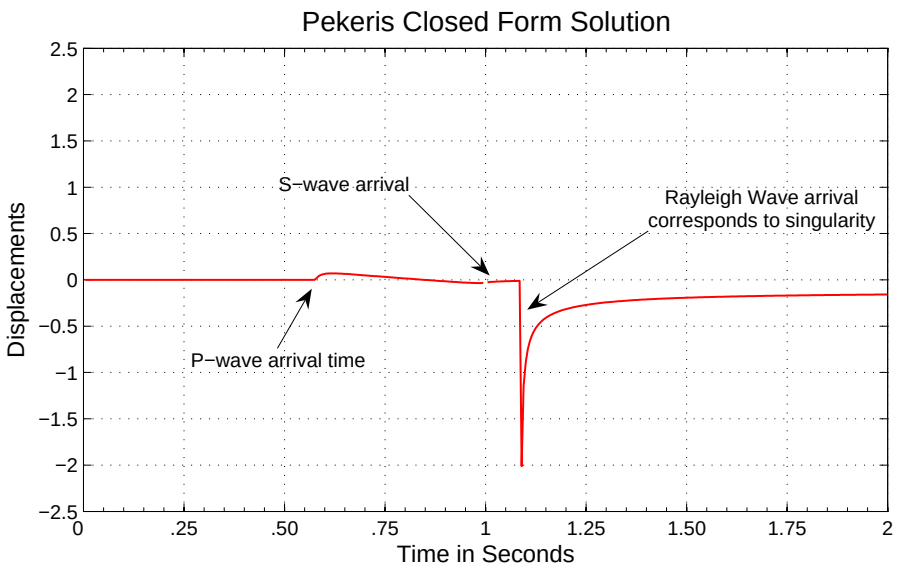


Figure 9. **Horizontal surface displacement $u_r(r, z, t)$ due to a concentrated point source directed downward on the surface. Poisson's ratio, ν, is $\frac{1}{4}$ with $\lambda = \mu$. The measurement is located one unit radially from the point of impact. P denotes the arrival time of compressional waves, and S of the shear wave. The Rayleigh wave propagates at the speed of the singularity.**

sidered the *anti-plane shear* version of the problem and employed cylindrical integral methods which take advantage of the wave propagation's symmetry. Both Pekeris and Achenbach used the Cagniard-de-Hoop method as an analytical technique for evaluating the multidimensional Fourier integrals, and obtained an inversion procedure for the displacement integrals which then could be solved using partial fraction decomposition. If the radial distance from the point source is unity and Poisson's ratio is $\frac{1}{4}$ with $\lambda = \mu$ then the vertical and horizontal displacement of a particle at the surface is displayed in Figures 8 and 9. Figure 8 depicts the vertical surface displacement $u_z(r, z, t)$ due to a concentrated point source directed downward on the surface while Figure 9 shows the corresponding horizontal surface displacement $u_r(r, z, t)$. In both figures the longitudinal wave arrival time is clearly seen. The shear wave arrival(though not as obvious) is also visible by a sudden change in the velocity of the wavefront as it passes the observation point. The Rayleigh wave arrival time is purely theoretical and coincides with the speed of a singularity that propagates along the free

surface. It is also noted that because of the functional singularity, the amplitude of the Rayleigh wave cannot be accurately determined. Technically, the singular point has an amplitude of infinity. Certainly, in practice the amplitude is finite. What Pekeris' closed-form solutions provides are the arrival times at which all three wave fronts pass an observation point. In this case that point is 1 meter radially from the source. This is extremely useful in helping to determine if the numerical FE model has the proper phase speeds within the computational area. So although the amplitude of the surface wave may not be easily expressed in terms of a definite magnitude, the arrival time of a disturbance propagating in the domain can be a key indicator of its presence in a FE model. The distribution of total energy for a single-element radiator among the shear, compressional, and Rayleigh waves are 25.8 percent for shear, 6.9 percent for compressional, and 67.4 percent for Rayleigh respectively as computed by Miller and Pursey(1955) [Ref. 29] for the idealized Poisson's ratio $\frac{1}{4}$ with $\lambda = \mu$, and experimentally observed by Wood(1968) [Ref. 30]. Figure 10 is a slice of SAFE-T's FE model which graphically shows the different wave propagation speeds as well as different displacement amplitudes among the three waves present in a seismic event. The anisotropic propagation in the $-z$ direction is because the elements are not square. This means that the shape of the elements must be "iso" if one wishes to model isotropic behavior.

3. Extension to Arbitrary Transient Source Wave Form

Transient sources such as the Heaviside function or the Dirac Delta function allow for the analytic solution as given above for Lamb's problem, but they are not realistic because of the non-physical singular impulse associated with them. However, the Heaviside function as a source and analytically computed by Pekeris can be extended by the principle of superposition and integral convolution to model any arbitrary transient waveform as an impact source. The necessity of such a convention

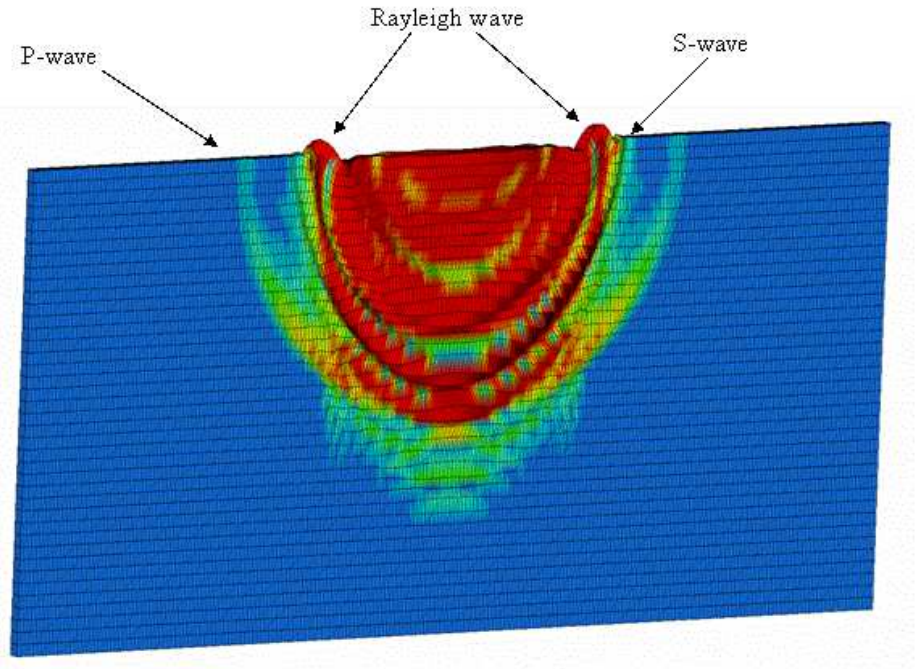


Figure 10. **This slice of SAFE-T's FE model graphically shows the different wave propagation speeds as well as different displacement amplitudes among the three waves present in a seismic event. The anisotropic propagation in the $-z$ direction is because the elements are not square.**

is seen in the fact that it is impossible to numerically model the instantaneous change of state associated with the Heaviside and Dirac Delta functions.

a. The Source

Physically speaking, sources can be similar to near-perfect adhesions. One example of this would be a weight dropped onto damp soil. Sources can also be simulated as near-perfect rebounds like a steel pellet onto some type of granite surface. Discussions on this matter with solid mechanical engineers along with the work of Rumph [Ref. 9] has led the author to select a single waveform which will be suitable for all models in this report. The source used in Rumph's experiment is a Haversine pulse. The Haversine waveform is produced by shifting the phase of the sine wave by 90 degrees and then adjusting the offset to set the baseline to zero. A

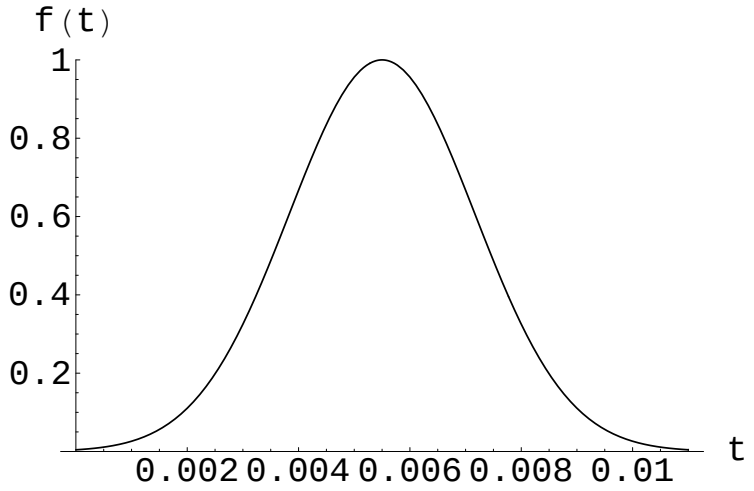


Figure 11. **Gaussian surface point source is modeled after Rumph's experimental Haversine source. The Haversine waveform is produced by shifting the phase of the sine wave by 90 degrees and then adjusting the offset to set the baseline to zero. This Gaussian has similar properties as the Haversine waveform source, is easier to implement into numerical code, and is everywhere differentiable.**

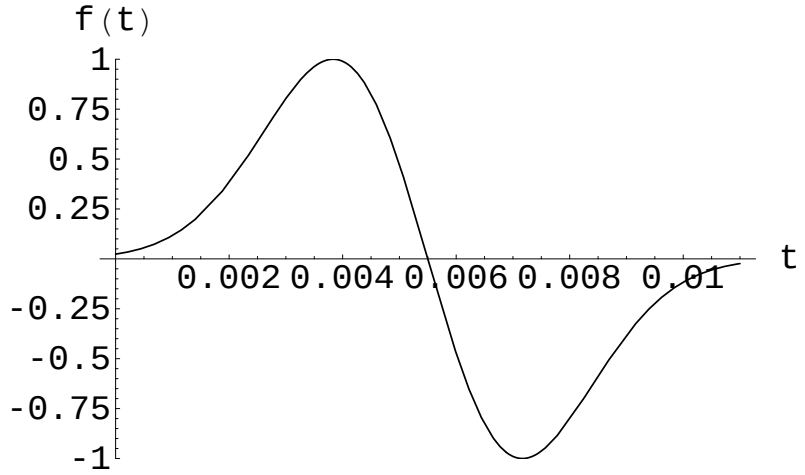


Figure 12. **Derivative of Gaussian surface source shows that the derivative is continuous everywhere.**

simpler source to use is a Gaussian source shown in figure 11 modeled with the same characteristics as Rumph's experimental Haversine pulse. It starts with zero slope, passes smoothly through a peak value and returns symmetrically to zero again. Since it is a Gaussian, it has a first derivative which is continuous everywhere, and allows for easy modifications to its amplitude, width, and location of peak (see figure 12) accomplished through the function

$$f(t) = (Amplitude)e^{-\left(\frac{t-peak}{width}\right)^2} \tag{II.100}$$

with derivative

$$\frac{df(t)}{dt} = -2\left(\frac{Amplitude}{(width)^2}\right)e^{-\left(\frac{t-peak}{width}\right)^2}(t-peak). \tag{II.101}$$

In addition, equations II.100 and II.101 allows the source to be tailored to specific characteristics. Namely, by adjusting the pulse width, one can change the power spectrum of the pulse generated. The power spectrum of the pulse in figure 12 is instrumental in determining the spatial resolution of the numerical grid which in turn, through stability requirements, enforces a limit to the time step that can be taken between each calculated solution.

In the present instance, the peak power of the pulse occurs at 599.675 hertz (see figure 13) which corresponds to a Rayleigh wavelength of one meter given parameters for a sandy medium (see Appendix C). The spatial resolution as measured by the largest distance between nodes in the numerical domain is less than $\frac{1}{12}$ of a meter, guaranteeing sufficient frequency for a Rayleigh wave. As there are other waves propagating in the elastic medium, they should also be weighed in mesh resolution considerations. Because the phase speeds for compressional waves and shear waves are greater than the Rayleigh wave speed, their corresponding wavelengths at the dominant frequency will be even greater than those of the Rayleigh wavelength guaranteing even better resolution.

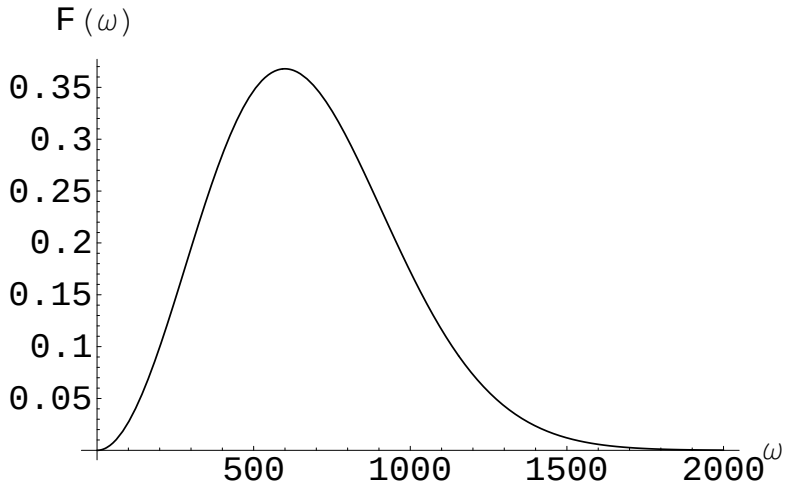


Figure 13. **The power spectrum of the derivative of Gaussian surface source reveals the dominant frequency.**

b. Convolution

This section outlines the development of a more useful analytic result. Starting with the specific problem treated by Pekeris, expressions for vertical and horizontal displacements appear below where τ represents the dimensionless quantity $\frac{c_s t}{r}$, and parameters k and κ are related as

$$k^2 = \frac{1}{\kappa^2} = \frac{3\tau^2 - 1}{2}. \tag{II.102}$$

The vertical displacement is expressed as

$$V(\tau) = \begin{cases} 0 & \tau < \frac{1}{\sqrt{3}} \\ -\frac{\pi}{96}\left\{6 - \frac{\sqrt{3\sqrt{3}+5}}{\sqrt{\gamma^2-\tau^2}} + \frac{\sqrt{3\sqrt{3}-5}}{\sqrt{\tau^2+\frac{\sqrt{3}}{4}-\frac{3}{4}}} - \frac{\sqrt{3}}{\sqrt{\tau^2-\frac{1}{4}}}\right\} & \frac{1}{\sqrt{3}} < \tau < 1 \\ -\frac{\pi}{48}\left\{6 - \frac{\sqrt{3\sqrt{3}+5}}{\sqrt{\gamma^2-\tau^2}}\right\} & 1 < \tau < \gamma \\ -\frac{\pi}{8} & \gamma < \tau \end{cases}$$

Applied Force Conversion Table			
Component	**Unit Step** $\boldsymbol{H(\tau)}$	**Impulse** $\boldsymbol{\delta(\tau)}$	**Arbitrary** $\boldsymbol{f(\tau)}$
Displacement (u_z or u_r)	$V(\tau)$	$\frac{dV(\tau)}{d\tau}$	$\tilde{V}(\tau)$
Velocity ($\dot{u}_z$ or $\dot{u}_r$)	$\frac{dV(\tau)}{d\tau}$	$\frac{d^2V(\tau)}{d\tau^2}$	$\frac{d\tilde{V}(\tau)}{d\tau}$
Acceleration ($\ddot{u}_z$ or $\ddot{u}_r$)	$\frac{d^2V(\tau)}{d\tau^2}$	$\frac{d^3V(\tau)}{d\tau^3}$	$\frac{d^2\tilde{V}(\tau)}{d\tau^2}$

Table I. **Conversion formulas for various source wave forms and measured quantities allow the use of integral convolution to find solutions for arbitrary transient source. Note: (1)** $\tilde{V}(\tau) = \frac{df(\tau)}{d\tau} * V(\tau)$ **and (2) For horizontal motion, replace** $V(\tau)$ **with** $H(\tau)$**.**

and horizontal displacement as

$$H(\tau) = \begin{cases} 0 & \tau < \frac{1}{\sqrt{3}} \\ -\frac{\tau}{16\sqrt{16}}\left\{6K(k) - 18\Pi(8k^2,k) + \frac{\Pi[(20-12\sqrt{3})k^2,k]}{(6-4\sqrt{3})^{-1}} + \frac{\Pi[(20+12\sqrt{3})k^2,k]}{(6+4\sqrt{3})^{-1}}\right\} & \frac{1}{\sqrt{3}} < \tau < 1 \\ -\frac{\tau\kappa}{16\sqrt{16}}\left\{6K(\kappa) - 18\Pi(8,\kappa) + \frac{\Pi[(20-12\sqrt{3}),\kappa]}{(6-4\sqrt{3})^{-1}} + \frac{\Pi[(20+12\sqrt{3}),\kappa]}{(6+4\sqrt{3})^{-1}}\right\} & \frac{1}{\sqrt{3}} < \tau < 1 \\ \text{Preceding} + \frac{\pi\tau}{24}\sqrt{\tau^2-\gamma^2} & \gamma < \tau \end{cases}$$

respectively with $\gamma = \frac{\sqrt{3+\sqrt{3}}}{2} \approx 1.08766$. $K(k)$ and $\Pi(n,k)$ are elliptic integrals of the first and third type. By convolving an arbitrary transient source function $f(t)$ and the functions given above using linear superposition over time, it is possible to find accurate comparisons for the FE model.

Since the time history of $f(t)$ is known, Mooney's [Ref. 6] technique can be followed and the use of the Duhamel integral with the quantities outlined in Table I are employed. Digital sampling of the source $f(t)$ and discrete convolution are used to calculate the response of the free surface and surrounding media due to the source pulse by the relation

$$\tilde{V}(\tau) = \frac{df(\tau)}{d\tau} * V(\tau) \qquad \text{(II.103)}$$

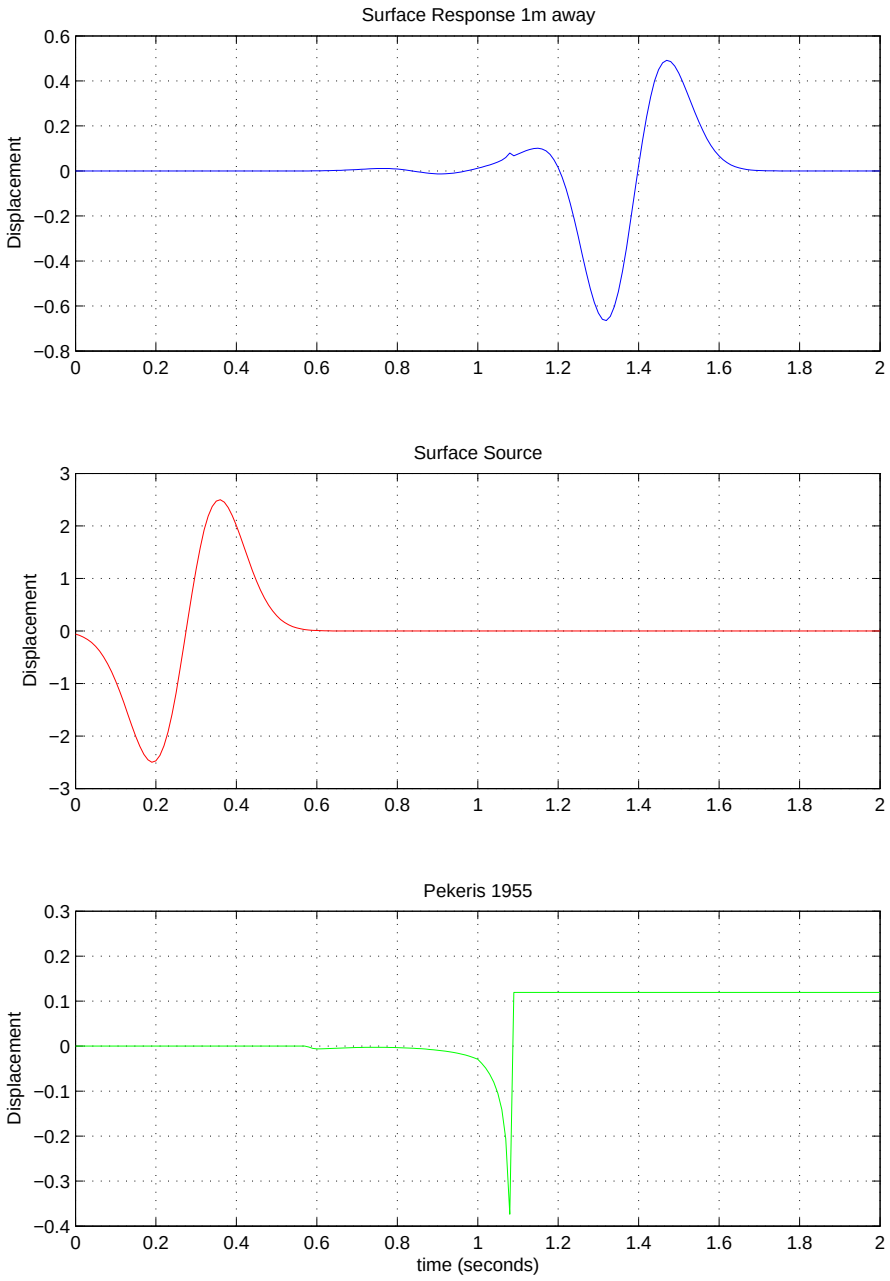


Figure 14. **Triple view of the two waveforms (2nd and 3rd graphs) which combine to demonstrate the response on the surface 1 meter away from the source (top graph).**

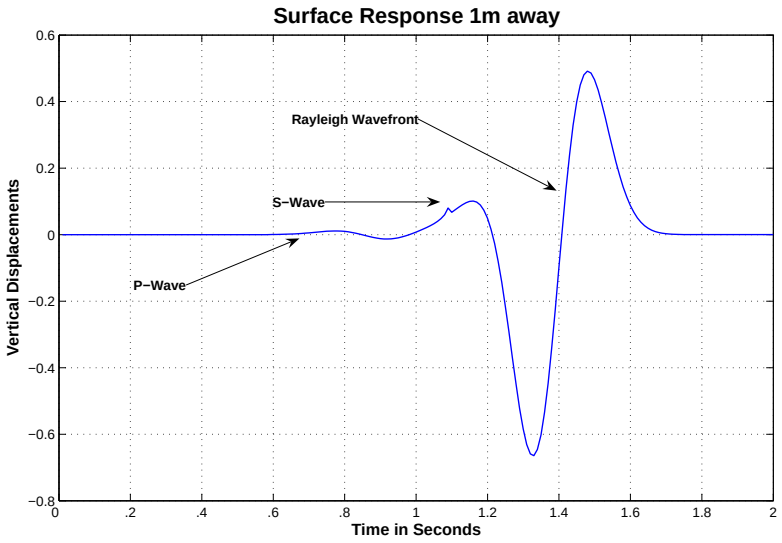


Figure 15. **Solution of derivative of a Gaussian source by linear superposition over time to determine the surface response 1 meter away from the source of excitation.**

or specifically,

$$\tilde{V}(\tau) = \sum_{i=0}^{N-1} \frac{df(\tau_i)}{d\tau} V(\tau - \tau_i). \tag{II.104}$$

The relationship between discrete and continuous convolution is well documented [Ref. 31]. In fact, the most important applications of the discrete convolution occur not by sampling periodic functions but rather by approximating continuous convolutions of waveforms [Ref. 32]. Figures 14 and 15 display convolutions made using the derivative of a Gaussian for $f(t)$ of the form of equation II.100.

C. NUMERICAL APPROACH

1. FE Modeling of Partial Differential Equations

A benchmark "analytic" solution to the well-posed partial differential equations and boundary conditions for a point/line loaded elastic half-space has been discussed. Except for a few simple cases it is nearly impossible to find analytic so-

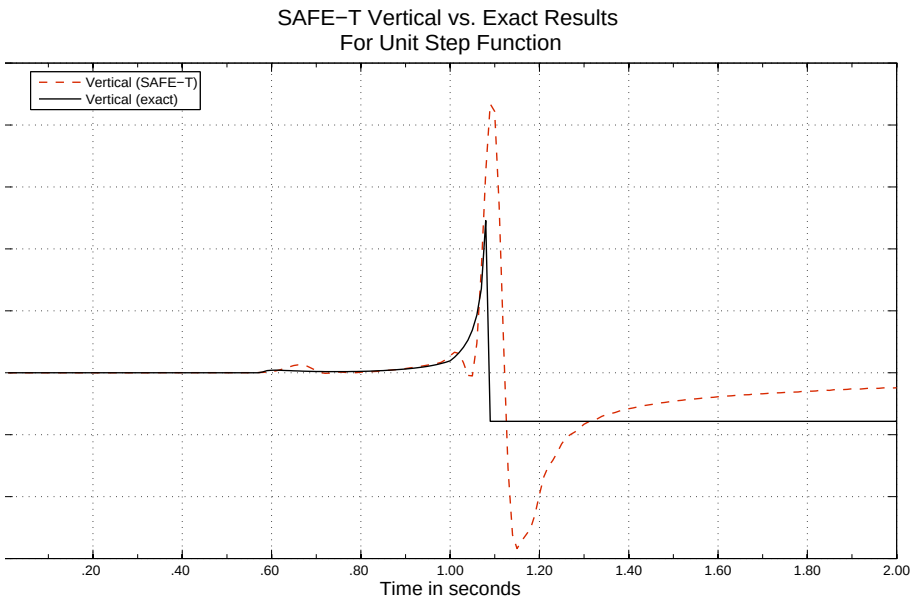


Figure 16. **Solid Adaptive Finite Element - Transient (SAFE-T) is used to model vertical displacement due to an instantaneous Heaviside change of state. 92 percent accuracy is achieved with 2,764,800 degrees of freedom. Δt time steps are taken at 0.0005 seconds. This particular calculation was done on a Compaq laptop computer with an AMD-64 Athlon processor and 2 gigabytes of memory. It took 17.78 hours of wall time with 3.05 hours of that being actual CPU time.**

lutions to the governing partial differential equations with nonsymmetric geometries and complex boundaries. The finite element method (FEM) offers an alternative to the somewhat limited class of problems for which analytic solutions can be found[Ref. 7]. By replacing the differential equation with an equivalent algebraic system of equations i.e., $Au = f$, it is possible to produce an approximate solution over a finite mesh of elements that models any given domain. This domain can include virtually any geometry including boundaries. The system of equations is assembled from all discrete *finite elements* of the domain, and solved to produce a solution over the entire domain. From the brief description given, it can be seen that the fundamental component of a FE solution is the production of a set of algebraic equations for each element. The functional form of the coefficients A and forcing term f in these equa-

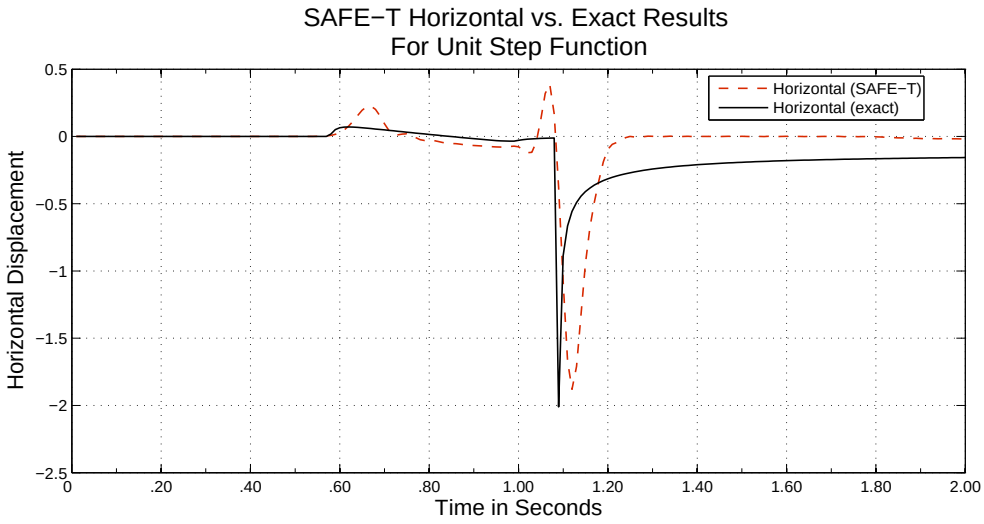


Figure 17. **Solid Adaptive Finite Element - Transient (SAFE-T) is used to model horizontal displacement due to an instantaneous Heaviside change of state. Again, roughly 92 percent accuracy is achieved with 2,764,800 degrees of freedom. Δt time steps are taken at 0.0005 seconds. This particular calculation was done on a Compaq laptop computer with an AMD-64 Athlon processor and 2 gigabytes of memory. It took 17.78 hours of wall time with 3.05 hours of that being actual CPU time.**

tions is identical for each element; only the numerical values of A and f differ from element to element. Thus, developing a FE formulation consists of developing the functional expression for A and f for a master set of element equations, which can then, like a master template, be numerically evaluated over and over again for each element in a mesh in order to generate the assembled system of algebraic equations. The theoretical development of the coefficients can be found in great detail in [Ref. 33].

a. Commercial FE Development Software

The principal software package, Prophlex, is a suite of developmental tool for creating customized finite element applications. Solid Adaptive Finite Element - Transient (SAFE-T) is a time dependent three dimensional finite element tool developed by the author using Prophlex for analysis of wave propagation in a

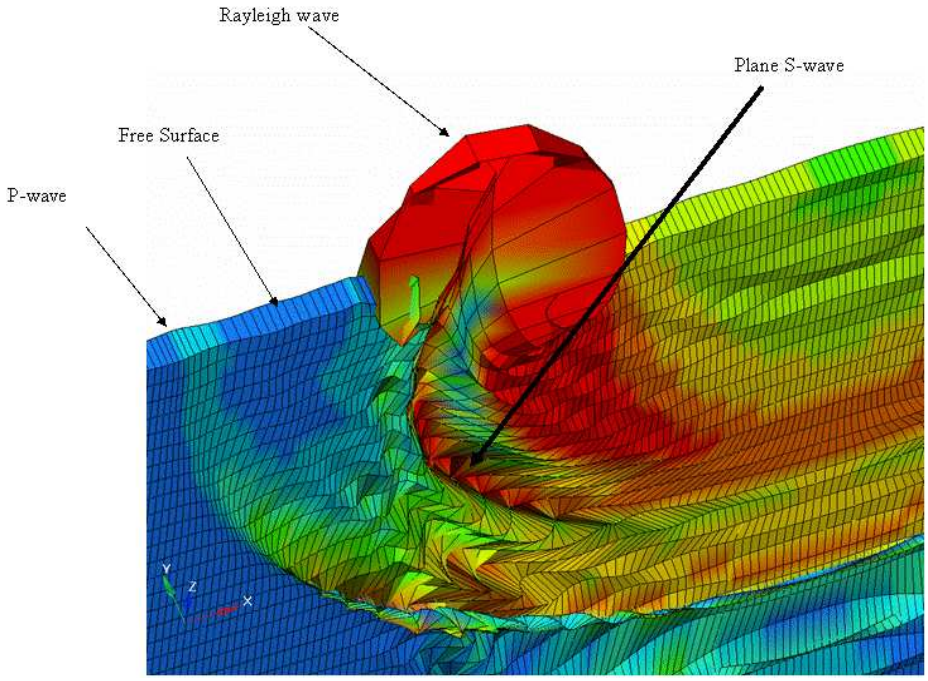


Figure 18. **This graphical depiction is the actual element by element deformation caused by a Rayleigh wave. It is modeled using SAFE-T's numerical output and postprocessed using the Altair Inc. Hyperview Postprocessing Suite. As the wave passes an observed point, all three waves (compressional, shear, and Rayleigh) can be seen. The values of λ and μ for this particular medium is equal to each other which makes ν 0.25.**

solid media. Prophlex is applicable to any system or process that can be mathematically formulated into a system of second-order PDEs. The FE mesh and other input data for the SAFE-T models are generated using Altair Engineering Inc. developed Hypermesh 7.0.

b. Examples

Figures 16 and 17 are examples of SAFE-T's numerical efforts in modeling such phenomena. Notice the early arrival of the p-wave in the SAFE-T results in figure 16 corresponds to the exact arrival time computed analytically with an error of about 8 percent. Figure 18 is a graphical depiction of a "scaled" element by element deformation of a Rayleigh wave. The actual amplitudes are on the order of microns.

2. Boundary and Initial Conditions

For any physical problem modeled by a PDE, many solutions are possible given the variety of geometries and diverse loads. To single out one particular solution requires formulation of physically realistic boundary and initial conditions which together make the problem both well-posed and useful [Ref. 34]. The FE method requires the same types of constraints as it, too, models physical problems that are themselves well-posed. As such three types of boundary conditions are needed in order to model the elastic half-space for this problem.

a. Displacement u_i is specified

$$u_i = U_i \tag{II.105}$$

b. Normal and Tangential Derivatives of displacements are specified (applied or free stress)

$$\tau_{ij}\hat{n} = t_i(x_i, t) \tag{II.106}$$

c. Normal Derivative of displacement is zero

$$\frac{\partial u_i}{\partial \hat{n}} = 0 \tag{II.107}$$

The initial conditions for this problem simply state that the medium is at rest.

3. Time Marching Scheme

The time marching scheme employed is based on the Generalized Newmark algorithm (GNpj) [Ref. 7]. This time marching scheme converges to the analytic

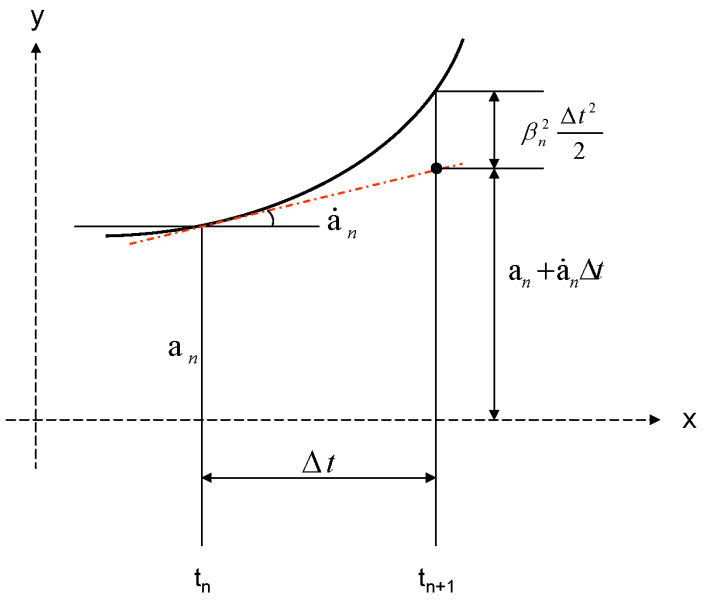


Figure 19. **The expansion of an unknown vector, say a, will be taken as a second degree polynomial with known values for a_n, $\dot{a}_n$, and $\ddot{a}_n$ at the beginning of each time step Δt.**

solution as the time step shrinks. Figure 19 is a graphical depiction of how succeeding solutions are calculated. The expansion of an unknown vector, say a, will be taken as a second degree polynomial with known values for a_n, $\dot{a}_n$, and $\ddot{a}_n$ at the beginning of time step Δt. The Lax-Equivalence Theorem states that the necessary and sufficient condition for a numerical method to be convergent are consistency and stability [Ref. 35]. Consistency simple means that as time intervals between calculating solutions is decreased, the truncation error between the numerical solution and the exact solution approaches zero. Convergence of a numerical method to an analytic solution implies that the numerical method is consistent, but the converse is not true. Consistency is not enough, but consistency with stability is enough. Zero-stability is concerned with the stability of the system in the limit as the time intervals between calculating the solution shrinks to zero. A built-in instability exists for initial value problems even in the limit as the time intervals approach zero in duration, but is mitigated by ensuring that the roots of the characteristic equation, or root condition, of the numerical method have absolute magnitudes that are less than or equal to unity.

According to Isaacson and Keller [Ref. 36], fact that the highest order of accuracy that can be expected from a k-step method is $2k$, so a single step, k=1, numerical method can be at most second order accurate. Use of a single step method conserves computer resources by calculating solutions with one pass of the computer processor. While it may be true that in seeking higher order the consistency condition is well satisfied, attempting to satisfy the condition for zero-stability becomes impossible. This barrier was first investigated by Germund Dahlquist [Ref. 37] and expounded upon by Lambert [Ref. 38]. Because of this barrier, the second-order single-step variant of GNpj called the Newmark Beta Method gives the highest order achievable in a single step. It is derived by making approximations for velocity and acceleration terms that come from third order Taylor Series expansion at the t^{n+1} time step.

$$u_i^{n+1} = u_i^n + \Delta t \frac{\partial u_i^n}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 u_i^n}{\partial t^2} + \frac{\Delta t^3}{6}\frac{\partial^3 u_i^n}{\partial t^3} + O(h^4) + \dots \tag{II.108}$$

The third derivative term is approximated by a backwards difference scheme leading to

$$u_i^{n+1} = u_i^n + \Delta t \frac{\partial u_i^n}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 u_i^n}{\partial t^2} + \beta \Delta t^2 \left(\frac{\partial^2 u_i^{n+1}}{\partial t^2} - \frac{\partial^2 u_i^n}{\partial t^2} \right) + O(\Delta t^4) + \dots \tag{II.109}$$

$$u_i^{n+1} = u_i^n + \Delta t \frac{\partial u_i^n}{\partial t} + \Delta t^2 \left(\frac{1}{2} - \beta \right) \frac{\partial^2 u_i^n}{\partial t^2} + \beta \Delta t^2 \frac{\partial^2 u_i^{n+1}}{\partial t^2} + O(\Delta t^4) + \dots \tag{II.110}$$

and now solving for the $\frac{\partial^2 u_i^{n+1}}{\partial t^2}$ term gives

$$\frac{\partial^2 u_i^{n+1}}{\partial t^2} = \frac{1}{\beta \Delta t^2}\left(u_i^{n+1} - u_i^n \right) - \frac{1}{\beta \Delta t}\frac{\partial u_i^n}{\partial t} - \left(\frac{1}{2\beta} - 1 \right) \frac{\partial^2 u_i^n}{\partial t^2} + O(\Delta t^2) + \dots \tag{II.111}$$

and similarly for the first derivative term we have

$$\frac{\partial u_i^{n+1}}{\partial t} = \frac{\gamma}{\beta \Delta t}\left(u_i^{n+1} - u_i^n \right) - \left(\frac{\gamma}{\beta} - 1 \right) \frac{\partial u_i^n}{\partial t} - \Delta t \left(\frac{\gamma}{2\beta} - 1 \right) \frac{\partial^2 u_i^n}{\partial t^2} + O(\Delta t^2) + \dots \tag{II.112}$$

The correct representations for derivative terms are now used to perform time marching in a one-step scheme. Thus, we accomplish a more economic use of computer resources as each time step can be solved in one pass of the computer processor.

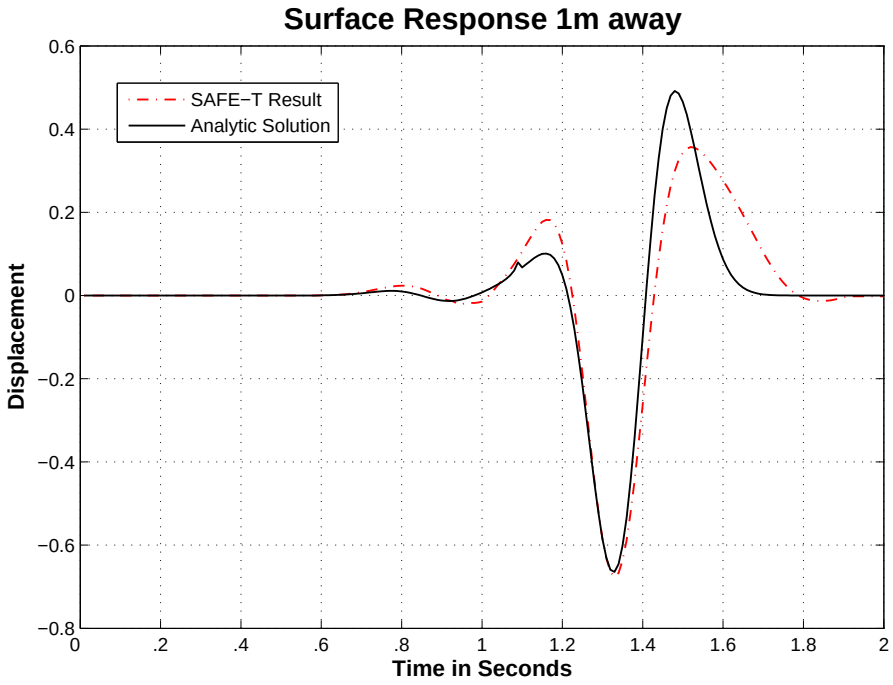


Figure 20. **SAFE-T vertical displacement results compared to the arbitrary source solution derived by discrete integral convolution. In this model an extended mesh is used to prevent unwanted body waves from interfering with the result.**

A transient numerical approach has been developed that incorporates initial and boundary conditions that makes the mathematical model both realistic and useful. By discrete integral convolution, an analytic benchmark has been computed to verify the accuracy of the FE method. Figure 20 is a comparison of SAFE-T's vertical displacement results to the analytic benchmark. This result is accomplished through the time-marching scheme we have just discussed. However, this is only half of the story. The other half centers about the difficult task of truncating the computational domain in such a way that it models an infinite half-space. We accomplish this by introducing a truncated elastic media via perfectly matched layers.

THIS PAGE INTENTIONALLY LEFT BLANK

III. TRUNCATED ELASTIC MEDIA

A. INTRODUCTION

A new unsplit 3D time dependent elastic PML PDE will be derived and converted to its weak (Galerkin) form for implementation into the finite element method. We have so far offered the finite element method as a numerical approach to the approximate solution of the elastic PDE with appropriate boundary conditions. This is well established as a reliable choice for problems that are finite. Infinite or semi-infinite problems are impossible without some way to absorb undesired reflections. Figure 21 is an example of the devastating effects of body wave reflections that makes reliable wave propagation modeling impossible. The graph represents the displacement history of a point on the free surface of the elastic half space. The non-PML response matches the PML response up to 40 Δt time steps, but soon reflection of body and surface waves corrupt the time histories. The reflections distorts the am-

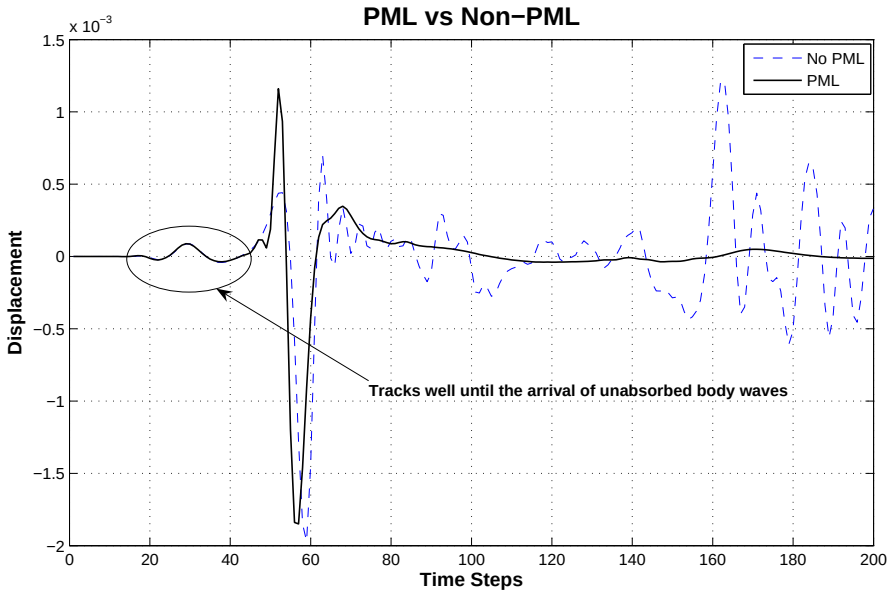


Figure 21. **The devastating effects of body waves make reliable wave propagation modeling impossible.**

plitude of the surface wave and renders it useless as a scatterer. The purpose of this chapter is to show that all reflections of body waves as well as reflected surface waves can be suppressed and absorbed by a transient PML boundary.

The objective of any PML method is to construct a new wave equation that causes waves to decay exponentially as they traverse the PML [Ref. 39]. This is accomplished most effectively by a complex change of variable for x_l as

$$x_l \longrightarrow \left(x_l + \frac{i}{\omega} \int_a^{x_l} f_l(\xi) d\xi \right) \tag{III.1}$$

where $f_l(x) \geq 0$ is the absorption function and a is the location of the PML interface. By replacing x_l in

$$\bar{u}_m(x_l, \omega) = A_m e^{i(x_l k_l - \omega t)} \tag{III.2}$$

we have

$$\bar{u}_m(x_l, \omega) = A_m e^{i(x_l k_l - \omega t)} e^{-\frac{k_l}{\omega} \int_a^{x_l} f_l(\xi) d\xi} \tag{III.3}$$

This transformation has the desired effect of introducing a purely real exponential term into the expression which acts to decay wave amplitude while the eigenvalues of the contained (computational domain) remain unchanged [Ref. 40]. If we let $\alpha(x_l)$ equal the non-zero real quantity used to suppress the wave such that

$$\alpha(x_l) = \frac{k_l}{\omega} \int_a^{x_l} f_l(\xi) d\xi \tag{III.4}$$

then the damping will be in the direction of x_l. For example, suppose one wanted to decrease a plane wave traveling in the x_1-direction. A new governing equation is required whose plane wave solution would have the form

$$\bar{u}_m(x_1, \omega) = A_m e^{i(x_1 k_1 - \omega t) - \alpha(x_1)} \tag{III.5}$$

where $\alpha(x_l)$ is a value that is nonzero only in the PML region and can be adjusted to produce a desired decay rate. The purpose of this complex coordinate stretching variable is to alter a wave's behavior as it traverses the PML region. Another way

to derive the complex coordinate stretching variable was introduced by Chew and Weedon [Ref. 16] as

$$\tilde{x}_j = \int_0^{x_j} F_j(\xi) d\xi, \quad (j = 1, 2, 3) \tag{III.6}$$

$$F_j(x) = 1 - i f_j(x) \tag{III.7}$$

Here f is an attenuation factor in the PML region and is zero within the computationally significant domain. Thus,

$$\alpha(x_j) = \int_0^{x_j} F_j(\xi) d\xi, \tag{III.8}$$

is used to produce

$$\tilde{x}_j = x_j - \alpha(x_j). \tag{III.9}$$

where the stretching coefficient along the prescribed axis x_j is a complex number. The resulting independent variable causes damping of wave fronts propagating in a prescribed direction, and is very efficient for wave absorption at the boundary of a numerical model [Ref. 41]. The complex coordinate stretching function, F, is continuous, and therefore, the stretch coordinate is smooth and the Fundamental Theorem of Calculus can be applied. It can be shown that if we differentiate equation III.6 with respect to x_j,

$$\frac{\partial \tilde{x}_j}{\partial x_j} = F_j(x_j) \tag{III.10}$$

which implies that

$$\frac{\partial}{\partial x_j} = \frac{1}{F_j} \frac{\partial}{\partial \tilde{x}_j}. \tag{III.11}$$

We now have a new differential operator for the governing equations that incorporate the PML properties of the media through its differential operator. The PDE becomes

$$\frac{1}{F_j} \frac{\partial \bar{\tau}_{ij}}{\partial \tilde{x}_j} = -\rho \omega^2 \bar{u}_i, \tag{III.12}$$

and the constitutive equations are

$$\bar{\tau}_{ij} = c_{ijkl} \bar{\epsilon}_{kl}, \tag{III.13}$$

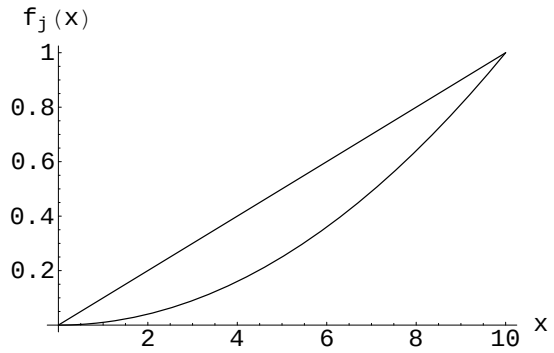


Figure 22. **Linear and nonlinear damping functions used in the PML.**

and

$$\bar{\epsilon}_{ij} = \frac{1}{2}\left[\frac{1}{F_j}\bar{u}_{i,j} + \frac{1}{F_i}\bar{u}_{j,i}\right] \tag{III.14}$$

where F has a complex value only in the absorbing PML region and is otherwise unity.

At the interface of the computational domain and the PML, the wave equations are identical so that any propagating wave will pass through the interface without generating reflected waves. In other words, there is an impedance match at the boundary of the unscaled computational domain and the PML. Care must be taken when choosing an appropriate F_j. It must be complex as defined in equation III.7 with an imaginary part related to the desired wave attenuation [Ref. 42]. It is also desirable to have the imaginary part of equation III.7 increase gradually relative to its position in the PML. This will provide zero attenuation at the interface yet gradually increase the attenuation as the wave travels in the direction of the PML x_j coordinate. Choosing $f(x)$ to be linear or nonlinear in the PML region are possible candidates for fulfilling the attenuation requirement. Figure 22 is a graph of an example of both the linear and nonlinear functions $f_j(x)$. In the linear case, $0 \leq f_j \leq 1$

$$f_j(x) = \begin{cases} \alpha_0 \left(\frac{l_{xj-PMLStart}}{L_{PML}}\right) & \text{if the plane wave is in the PML} \\ 0 & \text{if the plane wave is in the computational area} \end{cases} \tag{III.15}$$

In the nonlinear case, $0 \leq f_j \leq 1$

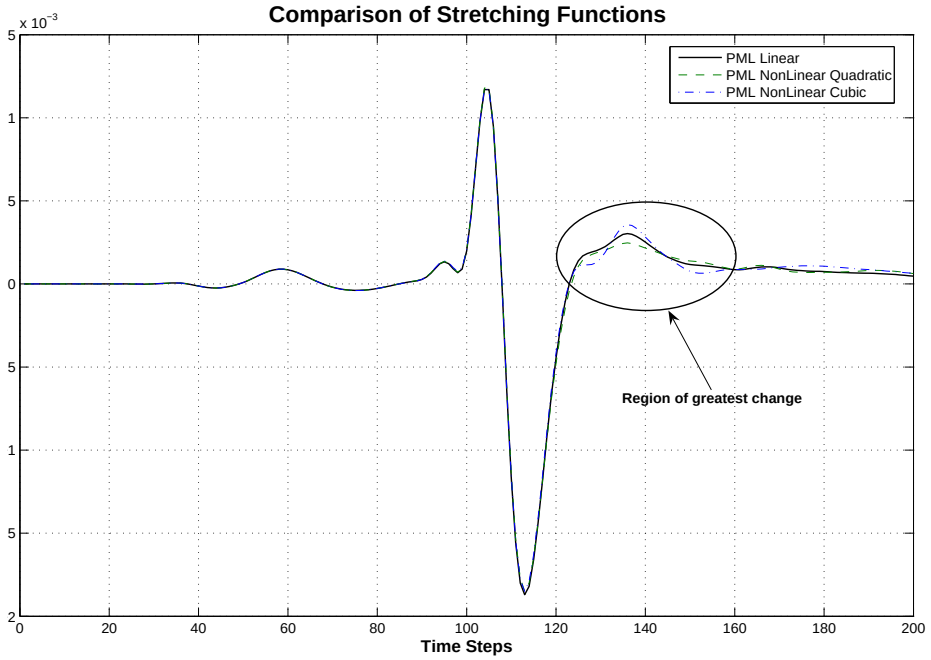


Figure 23. **Comparison of linear $\gamma = 1$, quadratic $\gamma = 2$, and cubic $\gamma = 3$ damping functions used within the PML region of the FE model. All three functions whether linear or non-linear provide excellent absorbtion with almost identical results.**

$$f_j(x) = \begin{cases} \alpha_0 \left(\frac{l_{xj-PMLStart}}{L_{PML}}\right)^{\gamma} & \text{if the plane wave is in the PML} \\ 0 & \text{if the plane wave is in the computational area} \end{cases} \tag{III.16}$$

where $PMLStart$ is where the PML region begins, L_{PML} is the total length of the PML region, γ is a non-linearity constant, α_0 is a damping constant, and l_{xj} is the length from the interface to the plane wave front. Figure 23 is a comparison of the effects of using linear, quadratic, and cubic functions as the damping functions within the PML region. All three functions whether linear or non-linear provide excellent absorbtion with almost identical results. Figure 24 analyzes the sensitivity of the pml to the damping constant, α_0.

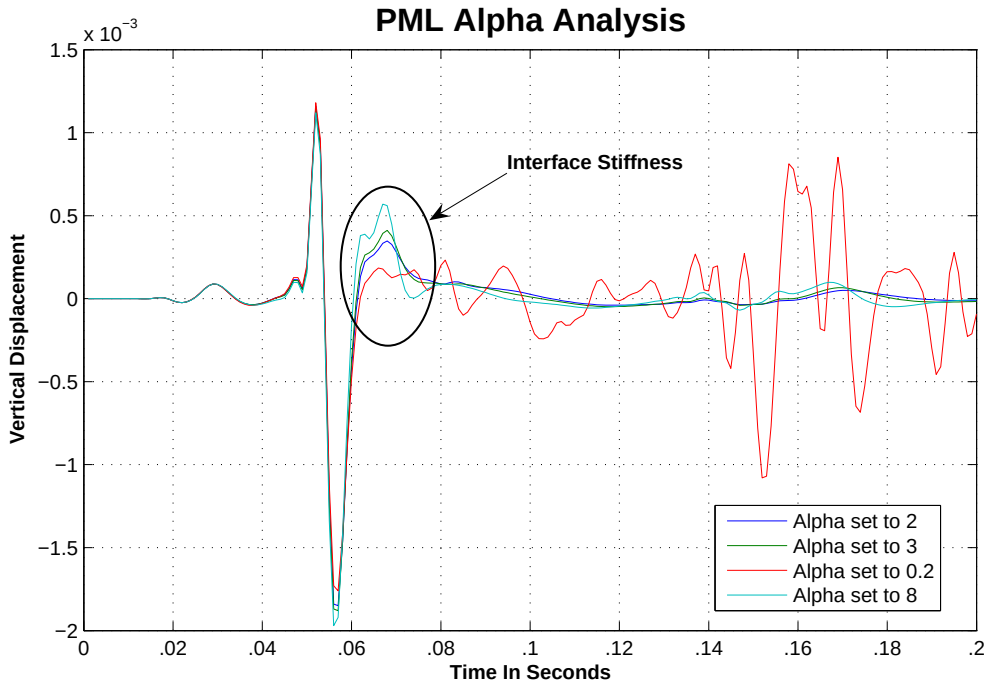


Figure 24. **Analysis of the PML to determine the sensitivity to the damping constant, α_0. The location of measurement is along the computational domain/PML interface. The bump circled is highlighting the reflected amplitude of the incident wave upon the boundary because of a stiffening of the interface. α_0 set to 2 provides the maximum absorption with minimal interface stiffness and incident reflection.**

B. 3D TRANSIENT PML EQUATION DERIVATIONS

Although PMLs are more complex to implement, especially in three dimensions, the methodology for formulating their equations are roughly the same. PMLs require only a finite number of nodes per wavelength. Between 4 and 8 nodes per wave length provide an excellent absorption of body waves, and do not lose efficiency at shallow angles. Most notably, PMLs have been shown to be very effective with surface waves. The reflections at the boundaries can be made arbitrarily small by increasing the thickness of the PML layer at the cost of additional computation[Ref. 41]. This cost, however, is usually well worth the extra resources required. With that in mind, we adopt a perfectly matched medium undergoing time-harmonic motion in

the absence of body forces, whose governing equations III.12, III.13, and III.14, are defined by the following:

$$\frac{\bar{F}_1(x_1)\bar{F}_2(x_2)\bar{F}_3(x_3)}{\bar{F}_j(x_j)}\bar{\tau}_{ij,j} = -\omega^2\rho\bar{F}_1(x_1)\bar{F}_2(x_2)\bar{F}_3(x_3)\bar{u}_i \tag{III.17}$$

$$\bar{\tau}_{ij} = \mathbf{C}_{ijkl}\bar{\epsilon}_{kl} \tag{III.18}$$

$$\bar{\epsilon}_{ij} = \frac{1}{2}\left[\frac{1}{\bar{F}_j(x_j)}\bar{u}_{i,j} + \frac{1}{\bar{F}_i(x_i)}\bar{u}_{j,i}\right]. \tag{III.19}$$

The stretching function, $\bar{F}_i(x_i)$, must possess special properties. It must be unity in the computational domain and complex otherwise with a real component that dampens evanescent waves and a complex component that dampens propagating waves in the PML. A detailed analysis of stretching functions can be found in [Ref. 11]. Choosing the stretching function to be of the form

$$\bar{F}_i(x_i) = \left([1 + f_i^e(x_i)] + \frac{c_s f_i^p(x_i)}{i\omega}\right) \tag{III.20}$$

where f^e and f^p are evanescent and propagating damping functions respectively, and c_s is the shear wave phase speed- used as the reference speed in the elastic medium, yields stretch matrices $\mathbf{A}$, $\hat{\mathbf{A}}$ and $\tilde{\mathbf{A}}$. These three matrices contain all the coordinate stretching information of the perfectly matched medium. When in the computational domain they are zero, ($\hat{\mathbf{A}}$ and $\tilde{\mathbf{A}}$), or identity ($\mathbf{A}$), but in the perfectly matched medium they attenuate both evanescent and propagating waves.

$$\mathbf{A} = \begin{pmatrix} (1+f_2^e)(1+f_3^e) & 0 & 0 \\ 0 & (1+f_3^e)(1+f_1^e) & 0 \\ 0 & 0 & (1+f_1^e)(1+f_2^e) \end{pmatrix}$$

$$\hat{\mathbf{A}} = \begin{pmatrix} c_s[f_2^p(1+f_3^e) + f_3^p(1+f_2^e)] & 0 & 0 \\ 0 & c_s[f_3^p(1+f_1^e) + f_1^p(1+f_3^e)] & 0 \\ 0 & 0 & c_s[f_1^p(1+f_2^e) + f_2^p(1+f_1^e)] \end{pmatrix}$$

$$\tilde{\mathbf{A}} = \begin{pmatrix} c_s^2 f_2^p f_3^p & 0 & 0 \\ 0 & c_s^2 f_3^p f_1^p & 0 \\ 0 & 0 & c_s^2 f_1^p f_2^p \end{pmatrix}.$$

The scaled or stretched governing partial differential equation in the frequency domain becomes

$$\mathbf{A}_{jm}\bar{\tau}_{ij,m} + \frac{\hat{\mathbf{A}}_{jm}\bar{\tau}_{ij,m}}{i\omega} - \frac{\tilde{\mathbf{A}}_{jm}\bar{\tau}_{ij,m}}{\omega^2}$$
$$= -\omega^2\rho\left([1+f_1^e] + \frac{c_s f_1^p}{i\omega}\right)\left([1+f_2^e] + \frac{c_s f_2^p}{i\omega}\right)\left([1+f_3^e] + \frac{c_s f_3^p}{i\omega}\right)\bar{u}_i \quad \text{(III.21)}$$

and contains within it, all the structure needed to implement a PML along any of the three coordinate directions in a rectangular system. Because the shear speed is chosen as the reference speed, shear and surface waves have near perfect absorbtion. Compressional waves which travel at greater phase speeds have minor reflections that can be made arbitrarily small by adjusting the thickness of the PML.

1. Integral and Inversion Techniques

a. Frequency Inversion

PML equations for the frequency domain are given in equation III.21. In order to use the PML method in the time domain, a transformation must occur. Since multiplication or division by the factor $i\omega$ in the frequency domain is equivalent to differentiation or integration respectively in the time domain, the equations are transformed into their time-dependent counterparts by application of equation II.36, the Fourier inversion formula [Ref. 14]. The application of the inverse transform assumes that $\bar{\tau}$ is zero when $\omega = 0$. With this assumption, equation III.21 is transformed from the frequency domain PML equation of motion to the time domain PML equation of motion

$$\mathbf{A}_{jm}\tau_{ij,m} + \hat{\mathbf{A}}_{jm}\int_0^t \tau_{ij,m}d\xi + \tilde{\mathbf{A}}_{jm}\int_0^t\int_0^\zeta \tau_{ij,m}d\xi d\zeta$$
$$= M\ddot{u}_i + D\dot{u}_i + Ku_i + L\int_0^t u_i d\xi \quad \text{(III.22)}$$

where following the convention used in [Ref. 14] in 2 dimensions and applying it to 3 dimensions yields a mass term, M, a damping term, D, a stiffness term, K,

and a time-integral term, L. These are somewhat unconventional from a continuum mechanics view, but naturally arise in a time-domain implementation for a PML, according to Zhao (1996), when field-splitting is avoided [Ref. 43].

$$M = \rho(1 + f_1^e)(1 + f_2^e)(1 + f_3^e)$$
$$D = \rho c_s[f_1^p(1 + f_2^e)(1 + f_3^e) + f_2^p(1 + f_1^e)(1 + f_3^e) + f_3^p(1 + f_1^e)(1 + f_2^e)]$$
$$K = \rho c_s^2[f_2^p f_3^p(1 + f_1^e) + f_1^p f_3^p(1 + f_2^e) + f_1^p f_2^p(1 + f_3^e)]$$
$$L = \rho c_s^3 f_1^p f_2^p f_3^p.$$

By making a substitution for the integrals, a 3-D transient elastic wave equation containing the PML parameters can be written in the complete form

$$\mathbf{A}_{jm}\tau_{ij,m} + \hat{\mathbf{A}}_{jm}\Phi_{ij,m} + \tilde{\mathbf{A}}_{jm}\Psi_{ij,m} = M\ddot{u}_i + D\dot{u}_i + Ku_i + LU_i \tag{III.23}$$

where

$$\Phi_{ij,m} = \int_0^t \tau_{ij,m}d\xi, \quad \Psi_{ij,m} = \int_0^t\int_0^\zeta \tau_{ij,m}d\xi d\zeta, \quad \Upsilon_i = \int_0^t u_i d\xi \tag{III.24}$$

b. Weak Formulation

Construction of a weak formulation of III.23 begins with factoring divergence terms and consolidating them on the LHS. This leads to

$$(\mathbf{A}_{jm}\tau_{ij} + \hat{\mathbf{A}}_{jm}\Phi_{ij} + \tilde{\mathbf{A}}_{jm}\Psi_{ij})_{,m} = M\ddot{u}_i + D\dot{u}_i + Ku_i + L\Upsilon_i \tag{III.25}$$

Proceeding in the usual manner, we multiply III.25 by a test function v_i and integrate over the domain, Ω:

$$[(\mathbf{A}_{jm}\tau_{ij} + \hat{\mathbf{A}}_{jm}\Phi_{ij} + \tilde{\mathbf{A}}_{jm}\Psi_{ij})_{,m}]v_i\partial\Omega = \int_\Omega (M\ddot{u}_i + D\dot{u}_i + Ku_i + L\Upsilon_i)v_i\partial\Omega \tag{III.26}$$

By *Green's Identity* (integrating by parts) we expand the terms on the LHS, and are left with

$$\int_\Omega [(\mathbf{A}_{jm}\tau_{ij} + \hat{\mathbf{A}}_{jm}\Phi_{ij} + \tilde{\mathbf{A}}_{jm}\Psi_{ij})v_i]_{,m}\partial\Omega - \int_\Omega (\mathbf{A}_{jm}\tau_{ij} + \hat{\mathbf{A}}_{jm}\Phi_{ij} + \tilde{\mathbf{A}}_{jm}\Psi_{ij})v_{i,m}\partial\Omega$$
$$= \int_\Omega (M\ddot{u}_i + D\dot{u}_i + Ku_i + L\Upsilon_i)v_i\partial\Omega \tag{III.27}$$

Finally, applying the divergence theorem, we derive a weak formulation of the 3-D transient elastic wave partial differential equation that incorporates the PML parameters. Thus,

$$\int_{\Gamma}[(\mathbf{A}_{jm}\tau_{ij}+\hat{\mathbf{A}}_{jm}\Phi_{ij}+\tilde{\mathbf{A}}_{jm}\Psi_{ij})v_i]\hat{n}_m d\Gamma-\int_{\Omega}(\mathbf{A}_{jm}\tau_{ij}+\hat{\mathbf{A}}_{jm}\Phi_{ij}+\tilde{\mathbf{A}}_{jm}\Psi_{ij})v_{i,m}\partial\Omega$$
$$=\int_{\Omega}(M\ddot{u}_i+D\dot{u}_i+Ku_i+L\Upsilon_i)v_i\partial\Omega \qquad \text{(III.28)}$$

where the first term in equation III.28 is a boundary surface integral.

2. Galerkin Surface Integral

Consider the surface integral equation III.28.

$$\int_{\Gamma}[(\mathbf{A}_{jm}\tau_{ij}+\hat{\mathbf{A}}_{jm}\Phi_{ij}+\tilde{\mathbf{A}}_{jm}\Psi_{ij})v_i]\hat{n}_m d\Gamma \qquad \text{(III.29)}$$

The integration is only over boundary surface elements. For 3-D brick elements, integration is over at most three faces of any brick. Because of the presence of the PML, this really leads to three possible conditions that may be prescribed on an element face as seen in Figure 25. Equation III.28 may be expanded to include the possibilities:

$$\int_{\Gamma}[(\mathbf{A}_{jm}\tau_{ij}+\hat{\mathbf{A}}_{jm}\Phi_{ij}+\tilde{\mathbf{A}}_{jm}\Psi_{ij})v_i]\hat{n}_m d\Gamma=\Gamma_D+\Gamma_N+\Gamma_T \qquad \text{(III.30)}$$

where

$$\Gamma_D=\int_{\Gamma_D}[(\mathbf{A}_{jm}\tau_{ij}+\hat{\mathbf{A}}_{jm}\Phi_{ij}+\tilde{\mathbf{A}}_{jm}\Psi_{ij})v_i]\hat{n}_m d\Gamma, \qquad \text{(III.31)}$$

$$\Gamma_N=\int_{\Gamma_N}[(\mathbf{A}_{jm}\tau_{ij}+\hat{\mathbf{A}}_{jm}\Phi_{ij}+\tilde{\mathbf{A}}_{jm}\Psi_{ij})v_i]\hat{n}_m d\Gamma, \qquad \text{(III.32)}$$

and

$$\Gamma_T=\int_{\Gamma_T}[(\mathbf{A}_{jm}\tau_{ij}+\hat{\mathbf{A}}_{jm}\Phi_{ij}+\tilde{\mathbf{A}}_{jm}\Psi_{ij})v_i]\hat{n}_m d\Gamma \qquad \text{(III.33)}$$

represent Dirichlet, Neumann, and Stress conditions, respectively.

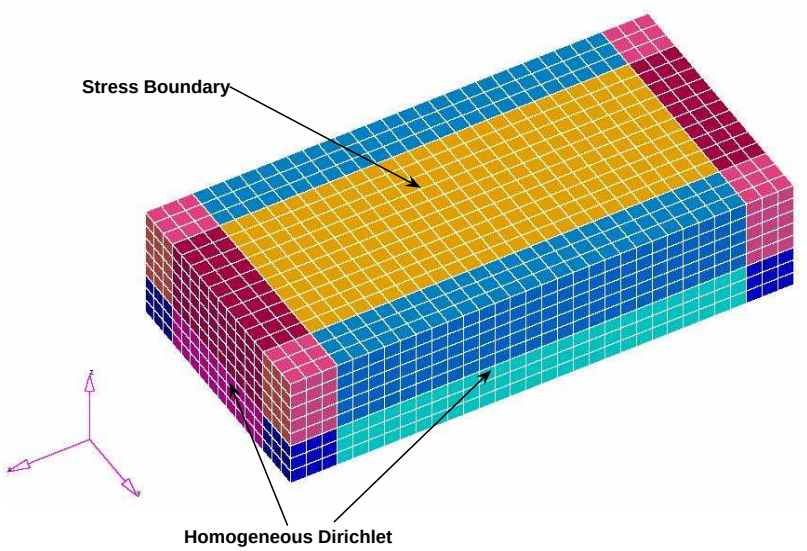


Figure 25. **Iso view of a FE model of a solid with Stress and Dirichlet boundaries. Any one of two conditions may be present on an element face.**

a. Condition: Dirichlet

The Dirichlet boundary condition assigns specific displacement (u_i) values on the boundary. Thus,

$$u_i = U_i. \tag{III.34}$$

However, this representation cannot be directly applied to the Dirichlet integral on the RHS of the expanded boundary integral (III.30). This is because the boundary integral is in terms of normal derivatives of stress instead of u_i. This condition can be circumvented by developing a penalty parameter formulation of the integral. It is accomplished by specifying that the difference between u_i and U_i be a small ϵ quantity such that

$$U_i - u_i = \epsilon(\mathbf{A}_{jm}\tau_{ij} + \hat{\mathbf{A}}_{jm}\Phi_{ij} + \tilde{\mathbf{A}}_{jm}\Psi_{ij})\hat{n}_m \tag{III.35}$$

$$\frac{1}{\epsilon}(U_i - u_i) = (\mathbf{A}_{jm}\tau_{ij} + \hat{\mathbf{A}}_{jm}\Phi_{ij} + \tilde{\mathbf{A}}_{jm}\Psi_{ij})\hat{n}_m \tag{III.36}$$

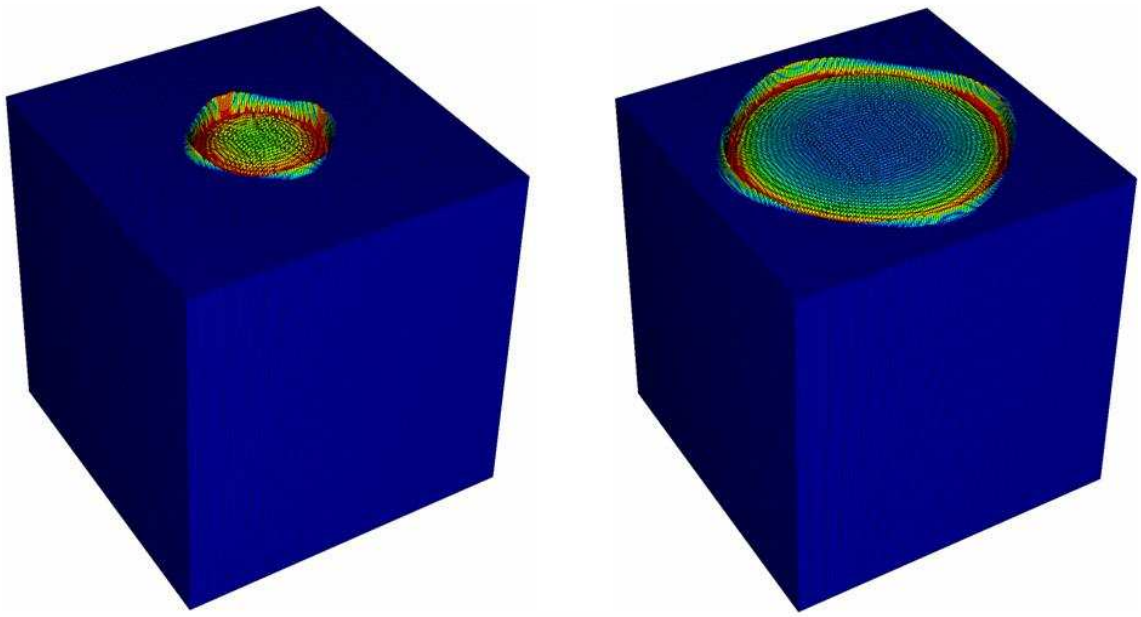


Figure 26. **Dual iso views of the FE model of a solid homogeneous cube. The boundaries of the cube are Dirichlet with a free surface directed upward. Notice the symmetry of wave propagation. The source is a Gaussian ignited at the center of the domain. Wave propagation is governed by the medium.**

where ϵ is the penalty parameter [Ref. 7]. Equation III.36 can be substituted directly into III.31 to give us a new representation for Γ_D:

$$\int_{\Gamma_D}[(\mathbf{A}_{jm}\tau_{ij} + \hat{\mathbf{A}}_{jm}\Phi_{ij} + \tilde{\mathbf{A}}_{jm}\Psi_{ij})v_i]\hat{n}_m d\Gamma = \frac{1}{\epsilon}\int_{\Gamma_D} U_i v_i d\Gamma - \frac{1}{\epsilon}\int_{\Gamma_D} u_i v_i d\Gamma \quad \text{(III.37)}$$

The value of ϵ is chosen to be extremely small. This is to make sure that when III.37 is applied to the final assembly matrix the two integrals on the RHS of III.37 will dominate all other algebraic terms in the equation. The ϵ domination will leave, in effect, only the two $\frac{1}{\epsilon}$ integrals in which case the coefficients cancel each other out. This leaves us with the condition defined by equation III.34. Typically, the guidelines for choosing ϵ is somewhere around eight to ten significant digits [Ref. 44].

b. Condition: Neumann

The Neumann boundary condition might be used along the exterior boundaries of the PML but if universally applied could lead to a non-unique solution of the governing equation. Some portion of the PML or free boundaries must be assigned Dirichlet conditions to ensure uniqueness of the numerical results.

c. Condition: Stress

The stress condition applies a specific value to normal derivatives of displacements at the bounded surface which may constitute a free surface when equal to zero or an applied known stress when inhomogeneous. We derive it by substituting

$$(\mathbf{A}_{jm}\tau_{ij} + \hat{\mathbf{A}}_{jm}\Phi_{ij} + \tilde{\mathbf{A}}_{jm}\Psi_{ij})\hat{n}_m = \begin{cases} \hat{t}_i(x_i,t), & \text{applied stress} \\ 0 & \text{stress free} \end{cases} \tag{III.38}$$

into the RHS of III.33 yielding

$$\int_{\Gamma_T} [(\mathbf{A}_{jm}\tau_{ij} + \hat{\mathbf{A}}_{jm}\Phi_{ij} + \tilde{\mathbf{A}}_{jm}\Psi_{ij})v_i]\hat{n}_m d\Gamma = \int_{\Gamma_T} \hat{t}_i(x_i,t)v_i d\Gamma \tag{III.39}$$

where $\hat{t}_i(x_i,t)$ specifies a specific time dependent source of stress.

3. Boundary Conditions

The Galerkin surface integral of this model defines two surface conditions: stress, and displacement. Because the free stress condition contributes nothing to the RHS of III.30, stress (where a non-zero forcing function is applied) and displacement are the only two boundary conditions that need be specifically calculated. The geometry of the half-space is very simple and requires no special treatment. As such, a fixed Dirichlet condition is suitable for the outer boundary of the PML boot which completely surrounds the computational domain except for the free surface of the half-space (see figure 25). The free surface source or stress traction requires more

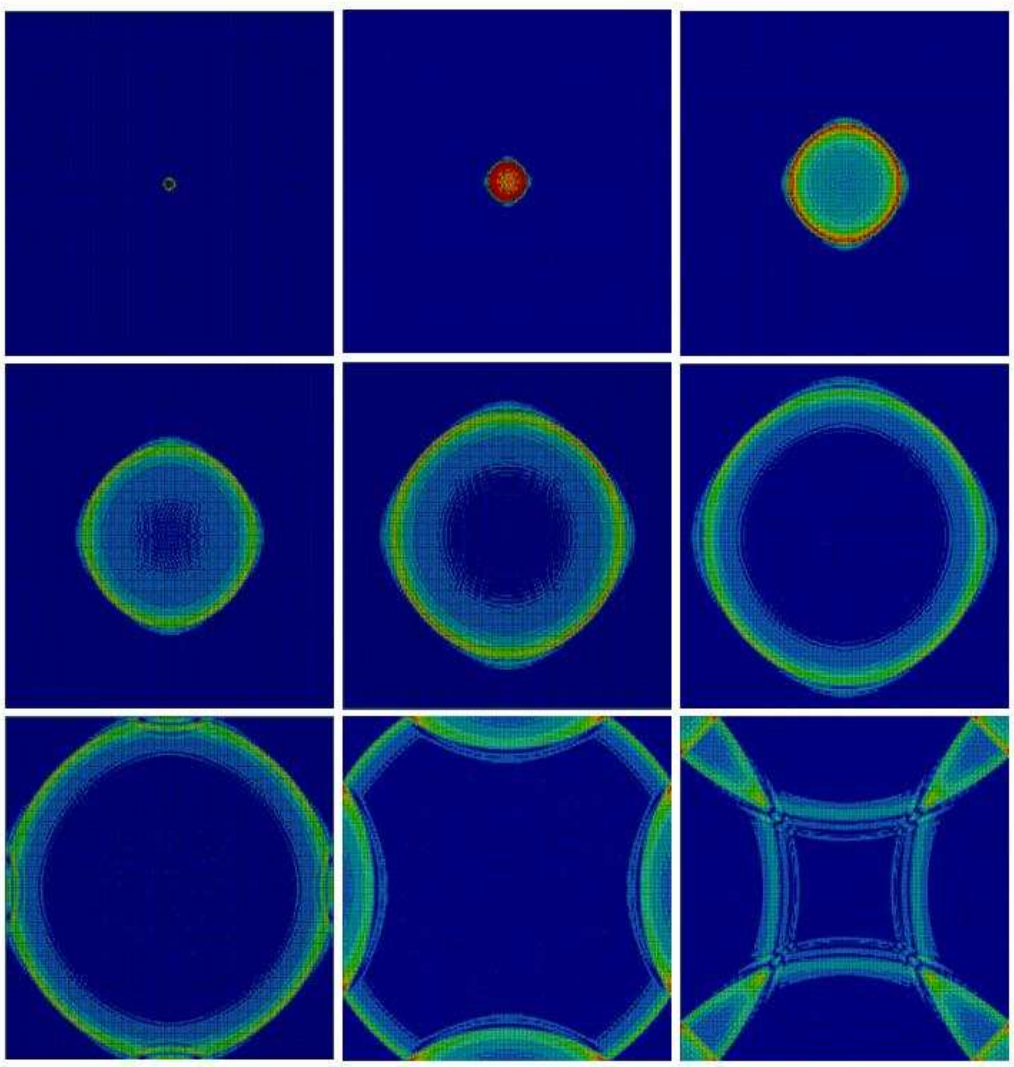


Figure 27. **Top view of a FE model of a solid cube with dirichlet boundaries. Notice the symmetry of wave propagation. The surface wave will impinge upon the boundary and return 180 degrees out of phase and travel back across the domain. This totally unphysical mathematical phenomenon is exactly what the PML is designed to solve. We say unphysical only in the sense that in an infinite half-space, waves travel outward and never return to the originating source.**

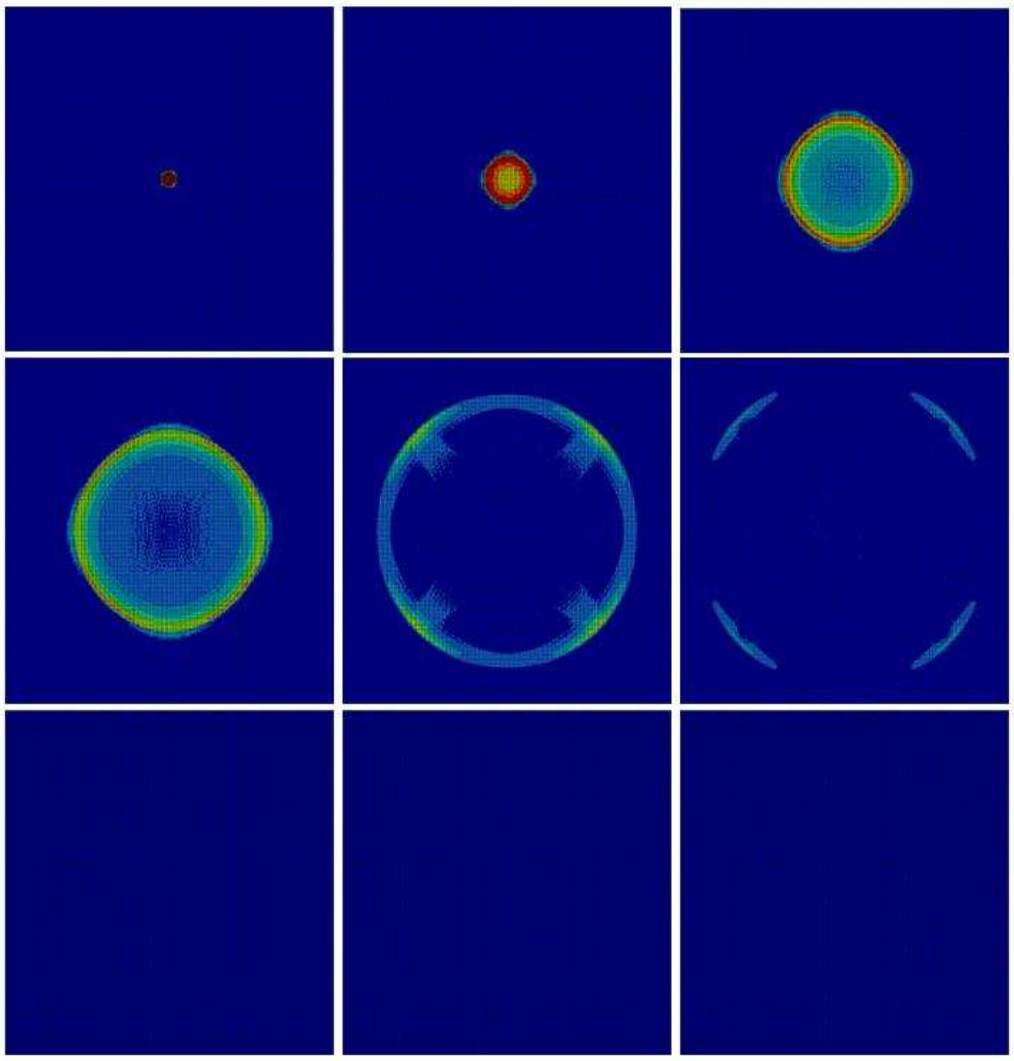


Figure 28. **Top view of a FE model of a solid cube with a PML boot truncated by Dirichlet boundaries. Again, notice the symmetry of wave propagation. The surface wave will impinge upon the boundary and be totally absorbed. This is exactly what the PML is designed to do. Waves travel outward and never return to the originating source.**

attention. The forcing function of the half-space problem is a point load directed downward. Figure 27 shows the reaction of waves excited by a point source impinging upon a Dirichlet boundary without a PML present. It represents one shaker that exerts a vertical traction only. This source is always placed within the computational domain away from the PML so that the stretch tensors are always zero or identity. Recalling the weak form of the elastic equation with PML parameters added we see that the only forcing term is the boundary surface integral, Γ_T, which is the normal component of the surface stress tensor. Figure 28 shows the reaction of waves excited by a point source impinging upon a Dirichlet boundary with a PML present. Note that to eliminate having a rectangular wavefront in figures 27 and 28 which is the result of grid dispersion, a finer mesh is used to produce a more cylindrical shape. A rectangular shape occurs when course meshes are used to model relatively fast wave phenomena. The graphs above clearly demonstrates the efficacy of the PML boundary.

4. Time Integration

The single-step recurrence relations derived in Chapter II, equations II.110, II.111, and II.112, are applied to the PML PDE, equation III.28. By substituting the approximations for the first, and second time derivatives into the weak form of the PDE and collecting unknown terms (primarily those at time t^{n+1}) on the LHS and collecting terms at time t^n and earlier on the RHS yields a discrete approximation to the solution of all primary variables. Note that stress and displacement boundary conditions are both known at t^{n+1} and, therefore, are placed on the RHS. Thus,

$$\int_\Omega \mathbf{S}^A_{jm}\tau^{n+1}_{ij}v_{i,m}\partial\Omega + \int_\Omega \left(tc_1M + tc_4D + K + \frac{L\Delta t}{2}\right)u_i^{n+1}v_i\partial\Omega + \frac{1}{\epsilon}\int_{\Gamma_D} u_i^{n+1}v_i d\Gamma$$
$$= \int_\Omega M\left[tc_1u_i^n + tc_2\frac{\partial u_i^n}{\partial t} + tc_3\frac{\partial^2 u_i^n}{\partial t^2}\right]v_i\partial\Omega + \int_\Omega D\left[tc_4u_i^n + tc_5\frac{\partial u_i^n}{\partial t} + tc_6\frac{\partial^2 u_i^n}{\partial t^2}\right]v_i\partial\Omega$$

$$- \int_\Omega \tilde{\mathbf{A}}_{jm} \Delta t^2 \sum_{l=1}^{n} ((n+1)-l) \tau_{ij}^l v_{i,m} \partial\Omega - \int_\Omega \hat{\mathbf{A}}_{jm} \Delta t \sum_{l=1}^{n} \tau_{ij}^l v_{i,m} \partial\Omega$$
$$- \int_\Omega L \Delta t \sum_{l=1}^{n} u_i^l v_i \partial\Omega + \int_{\Gamma_T} \hat{t}_i^{n+1}(x_i,t) v_i d\Gamma + \frac{1}{\epsilon} \int_{\Gamma_D} U_i^{n+1} v_i d\Gamma \tag{III.40}$$

where "time constants" 1 through 6 represent

$$tc_1 = \frac{1}{\beta \Delta t^2}$$
$$tc_2 = \frac{1}{\beta \Delta t}$$
$$tc_3 = \left(\frac{1}{2\beta} - 1 \right)$$
$$tc_4 = \frac{\gamma}{\beta \Delta t}$$
$$tc_5 = \left(\frac{\gamma}{\beta} - 1 \right)$$
$$tc_6 = \Delta t \left(\frac{\gamma}{2\beta} - 1 \right)$$

and

$$\mathbf{S}_{jm}^A = \left(\mathbf{A}_{jm} + \frac{\Delta t}{2} \hat{\mathbf{A}}_{jm} \right) \tag{III.41}$$

is a diagonal stretching matrix. The trapezoidal rule [Ref. 45] is used to approximate integrals with global truncation errors of $O(\Delta t^2)$:

$$\Phi_{ij}^{n+1} = \int_0^{t_{n+1}} \tau_{ij} d\xi \approx \Delta t \sum_{l=1}^{n} \tau_{ij}^l + \frac{\Delta t}{2} \tau_{ij}^{n+1} \tag{III.42}$$

$$\Psi_{ij}^{n+1} = \int_0^{t_{n+1}} \int_0^{\zeta} \tau_{ij} d\xi d\zeta \approx \Delta t^2 \sum_{l=1}^{n} ((n+1)-l) \tau_{ij}^l \tag{III.43}$$

$$\Upsilon_i^{n+1} = \int_0^{t_{n+1}} u_i d\xi \approx \Delta t \sum_{l=1}^{n} u_i^l + \frac{\Delta t}{2} u_i^{n+1} \tag{III.44}$$

C. STRAIN AND THE PML

At this point it is necessary to calculate the strain of a system in the PML. In order to mathematically capture the properties of the perfectly matched damping

media, a new stretching matrix will be defined. This matrix will allow the PML properties of the media to be combined with the classic strain equation in order to express a new strain equation. We begin with a diagonal matrix, say $\mathbf{S}$, which consists of the reciprocals of the stretching functions defined in III.20

$$\mathbf{S} = \begin{pmatrix} 1/\bar{F}_1(x_1) & 0 & 0 \\ 0 & 1/\bar{F}_2(x_2) & 0 \\ 0 & 0 & 1/\bar{F}_3(x_3) \end{pmatrix}. \tag{III.45}$$

Matrix $\mathbf{S}$ will be the mathematical vehicle used to transfer perfectly matched layer properties into the classic strain equation. Furthermore, the summation convention will be abandoned in certain cases below and in some cases matrix to matrix products will be found by multiplying term by term elements in each matrix to produce a product matrix. For example, the i, j component of $\mathbf{C} = \mathbf{AB}$ will become $\mathbf{A}_{i,j}\mathbf{B}_{i,j} = \mathbf{C}_{i,j}$. With that in mind; using matrix $\mathbf{S}$ above, we can rewrite the classic strain equation to yield

$$\bar{\epsilon}_{ij} = \frac{1}{2}(\bar{u}_{i,j}\mathbf{S} + \mathbf{S}\bar{u}_{j,i}). \tag{III.46}$$

Standard left and right multiplication by the inverse of the stretching matrix $\mathbf{S}$ yields,

$$\mathbf{S}^{-1}\bar{\epsilon}_{ij}\mathbf{S}^{-1} = \frac{1}{2}(\mathbf{S}^{-1}\bar{u}_{i,j}\mathbf{S}\mathbf{S}^{-1} + \mathbf{S}^{-1}\mathbf{S}\bar{u}_{j,i}\mathbf{S}^{-1}) \tag{III.47}$$

$$\mathbf{S}^{-1}\bar{\epsilon}_{ij}\mathbf{S}^{-1} = \frac{1}{2}(\mathbf{S}^{-1}\bar{u}_{i,j} + \hat{u}_{j,i}\mathbf{S}^{-1}). \tag{III.48}$$

Basu and Chopra(2003) [Ref. 14] make ample use of this technique for 1D and 2D strain manipulations. Here, for the 3D case, we make use of the technique to isolate the strain terms by the factor $i\omega$ with the substitution of the stretch function $\bar{F}_i(x_i)$, defined in equation III.20. By placing this into equation III.48 and abandoning summation over double indices in the expression below, we have

$$\begin{aligned} &\left([1 + f_j^e] + \frac{c_s f_j^p}{i\omega}\right)\left([1 + f_i^e] + \frac{c_s f_i^p}{i\omega}\right)\bar{\epsilon}_{ij} \\ &= \frac{1}{2}\left[\left([1 + f_i^e] + \frac{c_s f_i^p}{i\omega}\right)\bar{u}_{i,j} + \left([1 + f_j^e] + \frac{c_s f_j^p}{i\omega}\right)\bar{u}_{j,i}\right] \end{aligned} \tag{III.49}$$

Algebraic manipulations of the strain and displacement terms are grouped in powers of the frequency domain variables which await transformation to the time domain (again, there is no summation over the indices). Thus,

$$\left([1+f_i^e][1+f_j^e] + \frac{c_s\left(f_i^p[1+f_j^e]+f_j^p[1+f_i^e]\right)}{i\omega} - \frac{c_s^2 f_i^p f_j^p}{\omega^2} \right) \bar{\epsilon}_{ij}$$
$$= \frac{1}{2}\left[(1+f_i^e)\bar{u}_{i,j} + (1+f_j^e)\bar{u}_{j,i}\right] + \frac{c_s}{2i\omega}\left[f_i^p \bar{u}_{i,j} + f_j^p \bar{u}_{j,i}\right]. \qquad \text{(III.50)}$$

As pointed out by Basu and Chopra [Ref. 11], one can make use of the fact that transformations from the frequency domain to the time domain are simple when given terms that are multiples of $i\omega$. Further, if stretch matrices $\mathbf{B}$, $\hat{\mathbf{B}}$ and $\tilde{\mathbf{B}}$ are defined

$$\mathbf{B} = \begin{pmatrix} (1+f_1^e)(1+f_1^e) & (1+f_1^e)(1+f_2^e) & (1+f_1^e)(1+f_3^e) \\ (1+f_2^e)(1+f_1^e) & (1+f_2^e)(1+f_2^e) & (1+f_2^e)(1+f_3^e) \\ (1+f_3^e)(1+f_1^e) & (1+f_3^e)(1+f_2^e) & (1+f_3^e)(1+f_3^e) \end{pmatrix}$$

$$\hat{\mathbf{B}} = \begin{pmatrix} 2c_s[f_1^p(1+f_1^e)] & c_s[f_1^p(1+f_2^e)+f_2^p(1+f_1^e)] & c_s[f_1^p(1+f_3^e)+f_3^p(1+f_1^e)] \\ c_s[f_2^p(1+f_1^e)+f_1^p(1+f_2^e)] & 2c_s[f_2^p(1+f_2^e)] & c_s[f_2^p(1+f_3^e)+f_3^p(1+f_2^e)] \\ c_s[f_3^p(1+f_1^e)+f_1^p(1+f_3^e)] & c_s[f_3^p(1+f_2^e)+f_2^p(1+f_3^e)] & 2c_s[f_3^p(1+f_3^e)] \end{pmatrix}$$

$$\tilde{\mathbf{B}} = \begin{pmatrix} c_s^2 f_1^p f_1^p & c_s^2 f_1^p f_2^p & c_s^2 f_1^p f_3^p \\ c_s^2 f_2^p f_1^p & c_s^2 f_2^p f_2^p & c_s^2 f_2^p f_3^p \\ c_s^2 f_3^p f_1^p & c_s^2 f_3^p f_2^p & c_s^2 f_3^p f_3^p \end{pmatrix}.$$

the transformed strain equation in the time domain becomes

$$\mathbf{B}_{ij}\epsilon_{ij} + \hat{\mathbf{B}}_{ij}\int_0^t \epsilon_{ij}d\xi + \tilde{\mathbf{B}}_{ij}\int_0^t\int_0^\tau \epsilon_{ij}d\xi d\zeta$$
$$= \frac{1}{2}\left[(1+f_i^e)\hat{u}_{i,j} + (1+f_j^e)\hat{u}_{j,i}\right] + \frac{c_s}{2}\int_0^t \left[f_i^p\hat{u}_{i,j} + f_j^p\hat{u}_{j,i}\right]d\xi \qquad \text{(III.51)}$$

As remarked above, we will abandon the summation convention in favor of an element by element product convention. The strain is discretized in time and approximated

using the trapezoidal rule for the integral terms such that

$$\int_0^t \epsilon_{ij} d\xi \approx \Delta t \sum_{l=1}^{n} \epsilon_{ij}^l + \frac{\Delta t}{2} \epsilon_{ij}^{n+1} \tag{III.52}$$

$$\int_0^t \int_0^\tau \epsilon_{ij} d\xi d\zeta \approx \Delta t^2 \sum_{l=1}^{n} ((n+1) - l) \epsilon_{ij}^l \tag{III.53}$$

$$\int_0^t \left[f_i^p u_{i,j} + f_j^p u_{j,i} \right] d\xi \approx \Delta t \sum_{l=1}^{n} \left[f_i^p u_{i,j}^l + f_j^p u_{j,i}^l \right] + \frac{\Delta t}{2} \left[f_i^p u_{i,j}^{n+1} + f_j^p u_{j,i}^{n+1} \right] \tag{III.54}$$

Using the above approximations for the integral terms produces an equation for strain that incorporates the PML parameters. The $t^{(n+1)}$ time level strain is

$$\begin{aligned} \epsilon_{ij}^{n+1} \approx \frac{\mathbf{S}_{ij}^B}{2} \Bigg[& \left(F_i u_{i,j}^{n+1} + F_j u_{j,i}^{n+1} \right) + \ c_s \Delta t \sum_{l=1}^{n} \left(f_i^p u_{i,j}^l + f_j^p u_{j,i}^l \right) \\ & - 2\hat{\mathbf{B}}_{ij} \Delta t \sum_{l=1}^{n} \epsilon_{ij}^l - 2\tilde{\mathbf{B}}_{ij} \Delta t^2 \sum_{l=1}^{n} ((n+1) - l) \epsilon_{ij}^l \Bigg] \end{aligned} \tag{III.55}$$

where

$$\mathbf{S}_{ij}^B = \left(\mathbf{B}_{ij} + \frac{\Delta t}{2} \hat{\mathbf{B}}_{ij} \right)^{-1} \tag{III.56}$$

and

$$F_i(x_i) = \left([1 + f_i^e(x_i)] + \frac{c_s \Delta t}{2} f_i^p(x_i) \right). \tag{III.57}$$

Equation III.55 emphasizes a change in material parameters when used to derive stress. Since stress is proportional to strain, the perfectly matched media can thus be interpreted as a medium which exhibits inhomogeneous elastic properties. Note that outside the PML the stretching functions $f_i^e(x_i)$ and $f_i^p(x_i)$ are identically zero and the above strain tensor collapses to its classic form. When the above strain is placed into the stress-strain equation, we have

$$\tau_{ij}^{n+1} = \lambda \mathbf{S}_{kk}^B F_k u_{k,k}^{n+1} \delta_{ij} + \mu \mathbf{S}_{ij}^B \left[F_i u_{i,j}^{n+1} + F_j u_{j,i}^{n+1} \right] + \chi_{ij} \tag{III.58}$$

where the additive term χ_{ij} is defined as

$$\begin{aligned} \chi_{ij} = & \lambda \mathbf{S}_{kk}^B \left[c_s \Delta t \sum_{l=1}^{n} f_k^p u_{k,k}^l - \hat{\mathbf{B}}_{kk} \Delta t \sum_{l=1}^{n} \epsilon_{kk}^l - \tilde{\mathbf{B}}_{kk} \Delta t^2 \sum_{l=1}^{n} ((n+1) - l) \epsilon_{kk}^l \right] \delta_{ij} \\ & + \mu \mathbf{S}_{ij}^B \left[c_s \Delta t \sum_{l=1}^{n} \left(f_i^p u_{i,j}^l + f_j^p u_{j,i}^l \right) - 2\hat{\mathbf{B}}_{ij} \Delta t \sum_{l=1}^{n} \epsilon_{ij}^l - 2\tilde{\mathbf{B}}_{ij} \Delta t^2 \sum_{l=1}^{n} ((n+1) - l) \epsilon_{ij}^l \right] \end{aligned} \tag{III.59}$$

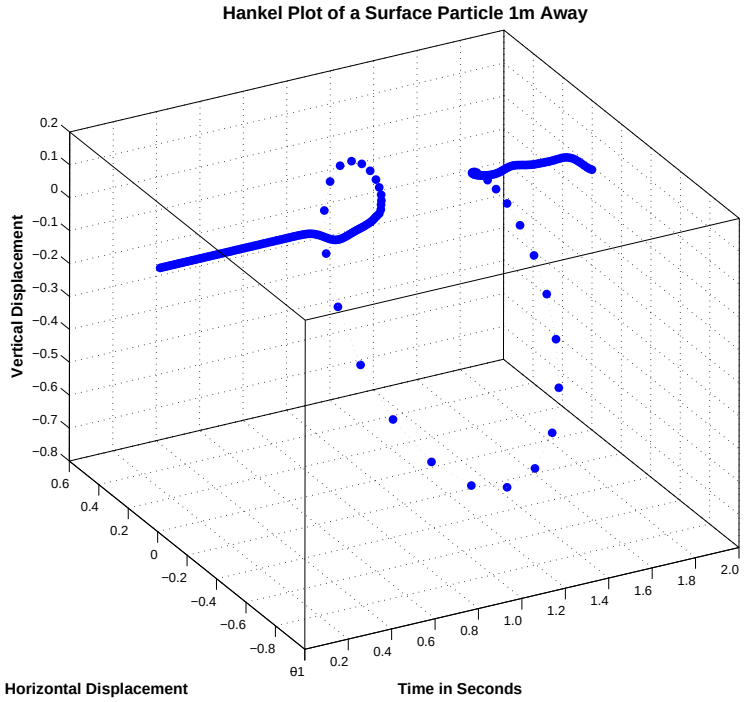


Figure 29. **This Hankel plot is of a particle on the surface of a half-space r distance away from the source. It clearly shows that as the wave moves across the surface, a particle moves in a circular or elliptic pattern in the direction of propagation which is characteristic of Rayleigh waves.**

The current stress, τ_{ij}^{n+1}, can now be used in the weak Galerkin formulation. Notice again that inside the computational domain, the value of χ_{ij} is zero as expected since inside the computational domain there is no artificial damping. If the value of χ_{ij} were not zero outside the PML region of the model, damping or exponential growth would be present that might cause unrealistic growth or decay in regions of the domain of interest.

D. FINAL GALERKIN FORM

The Galerkin (weak) form of the problem is arranged by placing all terms with implicit components on the LHS and explicit components on the RHS. This form of the equation provides all the information needed to set up the FE model in SAFE-T using the Prophlex kernel. Figures 29 and 2 are results obtained by SAFE-T using ideal material properties. The problem is idealized because in this instance the material properties λ and μ are set equal to each other.

$$\begin{aligned}
&\int_\Omega \mathbf{S}^A_{jm}\left(\lambda \mathbf{S}^B_{kk} F_k u^{n+1}_{kk}\delta_{ij} + \mu \mathbf{S}^B_{ij}\left[F_i u^{n+1}_{i,j} + F_j u^{n+1}_{j,i}\right]\right) v_{i,m}\partial\Omega \\
&+ \int_\Omega \left(\kappa_1 + K + \frac{L\Delta t}{2}\right) u_i^{n+1} v_i \partial\Omega + \frac{1}{\epsilon}\int_{\Gamma_D} u_i^{n+1} v_i d\Gamma \\
&= \int_\Omega \left(\kappa_1 u_i^n + \kappa_2 \frac{\partial u_i^n}{\partial t} + \kappa_3 \frac{\partial^2 u_i^n}{\partial t^2}\right) v_i \partial\Omega - \int_\Omega \mathbf{S}^A_{jm}\chi_{ij} v_{i,m}\partial\Omega \\
&- \int_\Omega L\Delta t \sum_{l=1}^{n} u_i^l v_i \partial\Omega - \int_\Omega \tilde{\mathbf{A}}_{jm}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau^l_{ij} v_{i,m}\partial\Omega \\
&- \int_\Omega \hat{\mathbf{A}}_{jm}\Delta t \sum_{l=1}^{n}\tau^l_{ij} v_{i,m}\partial\Omega + \int_{\Gamma_T} \hat{t}_i^{n+1}(x_i,t) v_i d\Gamma + \frac{1}{\epsilon}\int_{\Gamma_D} U_i^{n+1} v_i d\Gamma
\end{aligned} \tag{III.60}$$

where

$$\kappa_1 = tc_1 M + tc_4 D \tag{III.61}$$

$$\kappa_2 = tc_2 M + tc_5 D \tag{III.62}$$

$$\kappa_3 = tc_3 M + tc_6 D \tag{III.63}$$

Simplifying and collecting terms yields the final compact form of the equation:

$$\begin{aligned}
&\int_\Omega \mathbf{S}^A_{jm}\left(\lambda \mathbf{S}^B_{kk} F_k u^{n+1}_{kk}\delta_{ij} + \mu \mathbf{S}^B_{ij}\left[F_i u^{n+1}_{i,j} + F_j u^{n+1}_{j,i}\right]\right) v_{i,m}\partial\Omega \\
&+ \int_\Omega \left(\kappa_1 + K + \frac{L\Delta t}{2}\right) u_i^{n+1} v_i \partial\Omega + \frac{1}{\epsilon}\int_{\Gamma_D} u_i^{n+1} v_i d\Gamma \\
&= \int_\Omega \left(\kappa_1 u_i^n + \kappa_2 \frac{\partial u_i^n}{\partial t} + \kappa_3 \frac{\partial^2 u_i^n}{\partial t^2} - L\Delta t \sum_{l=1}^{n} u_i^l\right) v_i \partial\Omega \\
&- \int_\Omega \left(\mathbf{S}^A_{jm}\chi_{ij} + \hat{\mathbf{A}}_{jm}\Delta t \sum_{l=1}^{n}\tau^l_{ij} + \tilde{\mathbf{A}}_{jm}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau^l_{ij}\right) v_{i,m}\partial\Omega \\
&+ \int_{\Gamma_T} \hat{t}_i^{n+1}(x_i,t) v_i d\Gamma + \frac{1}{\epsilon}\int_{\Gamma_D} U_i^{n+1} v_i d\Gamma.
\end{aligned} \tag{III.64}$$

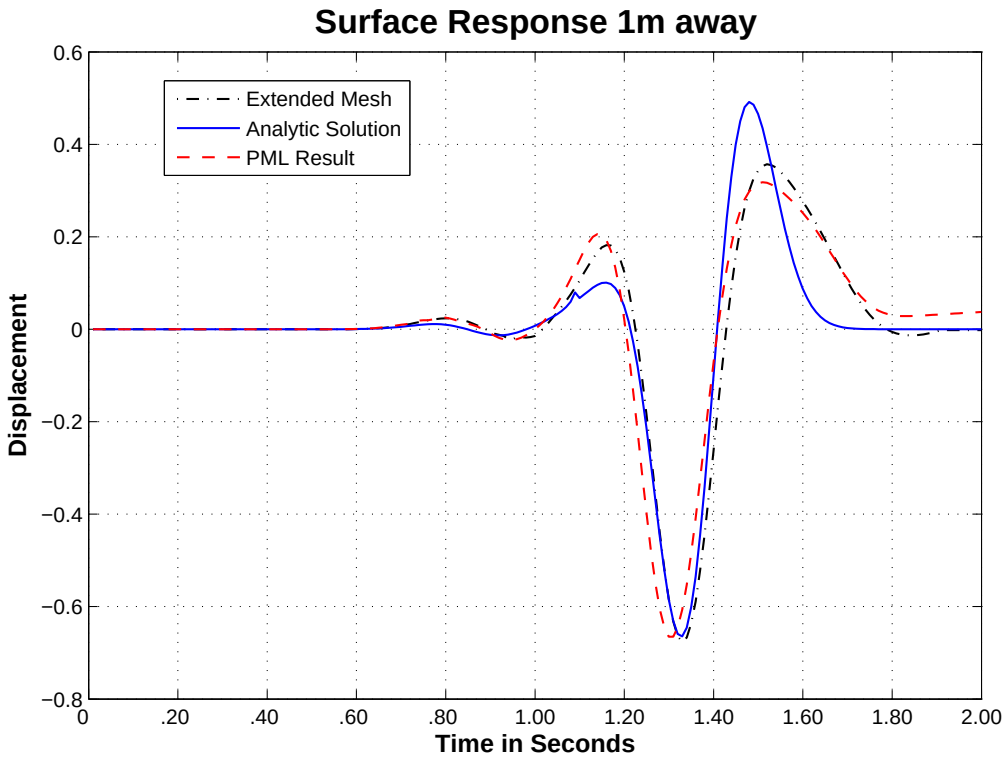


Figure 30. **A demonstration of the remarkable effects of the PML shows that, although not identical to the analytic solution, the result is very close to the extended mesh solution. The graph shows that the PML has a slight effect on the results obtained within the computational domain. This is due, in part, to the fact that the P-wave is not perfectly matched.**

The final Galerkin form is implemented into SAFE-T by way of Prophlex through FORTRAN and C++ subroutines.

E. SAFE-T RESULTS: PML EFFECTIVENESS

The true benefit of the PML method is its ability to allow the domain to be smaller than using an extended mesh. Extended meshes are costly. For example, if one wanted to use an extended mesh to model elastic wave phenomena in sand, you would need a domain greater than 800 meters to model 1 second of Rayleigh

activity. For 1D problems that would require 6400 elements. In 3D, it is triple that amount. The reason is that the P-wave travels about 1600 meters per second in sand while the Rayleigh wave moves only 95 meters per second. The remaining figures and tables of this chapter are an analysis of the thickness of a PML truncating an elastic computational domain.

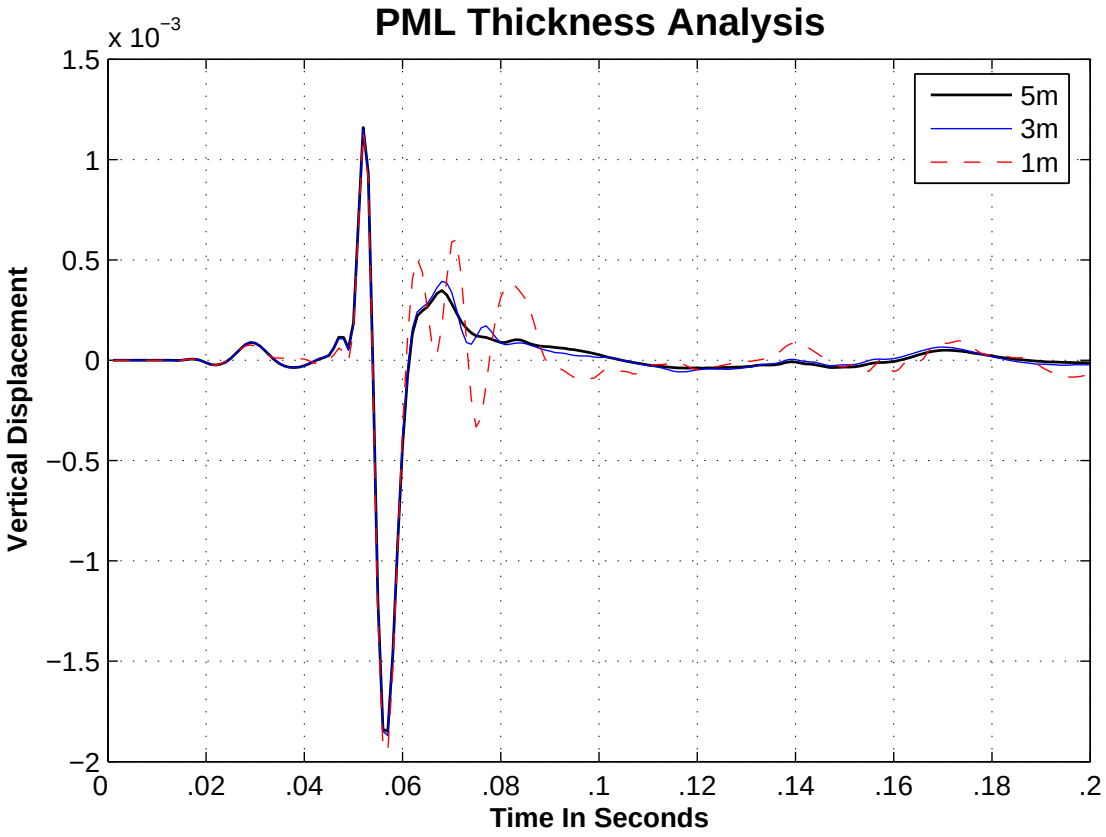


Figure 31. **SAFE-T PML Thickness Analysis (Vertical Displacements) shows that a PML that is 3m in depth is very similar to a PML 5m in depth. This resource savings is not trivial. The source is the derivative of a Gaussian. It has a dominant frequency of 599.675 Hz. The characteristic wavelength is 1 meter with a mesh density of 8 nodes per meter before refinement. The Δt time step of 0.0005 is well within the CFL stability criteria.**

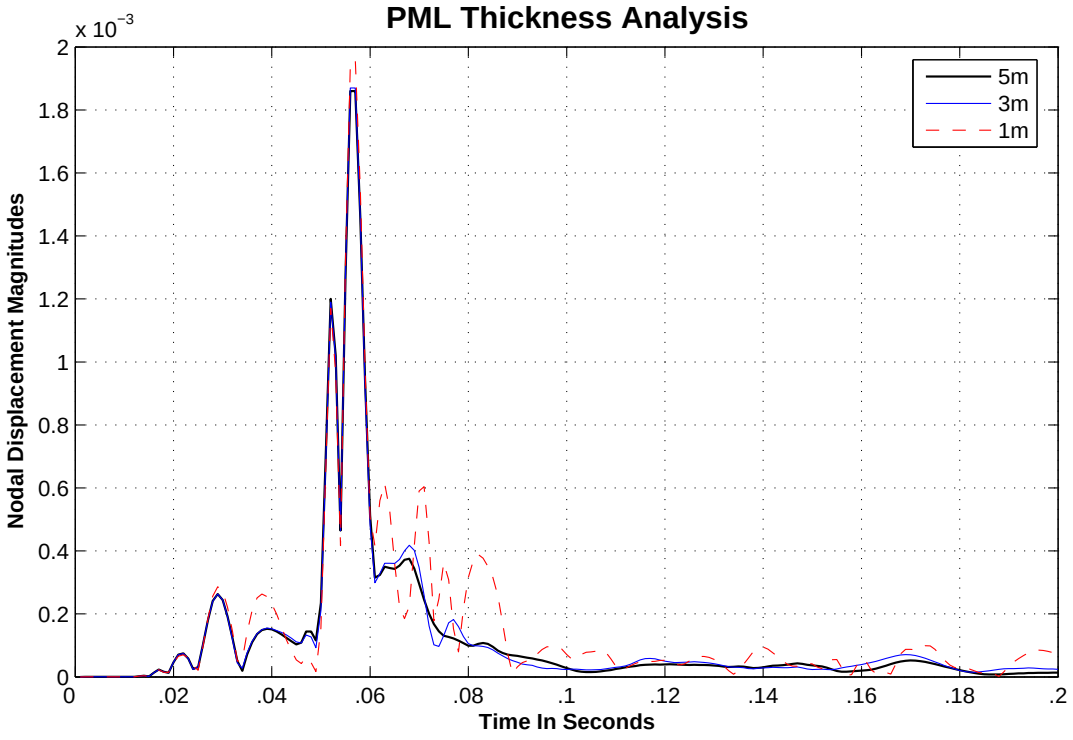


Figure 32. **SAFE-T PML Thickness Analysis of (Total Nodal Displacement Magnitudes). The bump at about .04 seconds shows the early arrival of reflected waves from the fixed boundary truncating the PML. Again, a PML that is 3m in depth is very similar to a PML 5m in depth. The characteristic wavelength is 1 meter with a mesh density of 8 nodes per meter before refinement.**

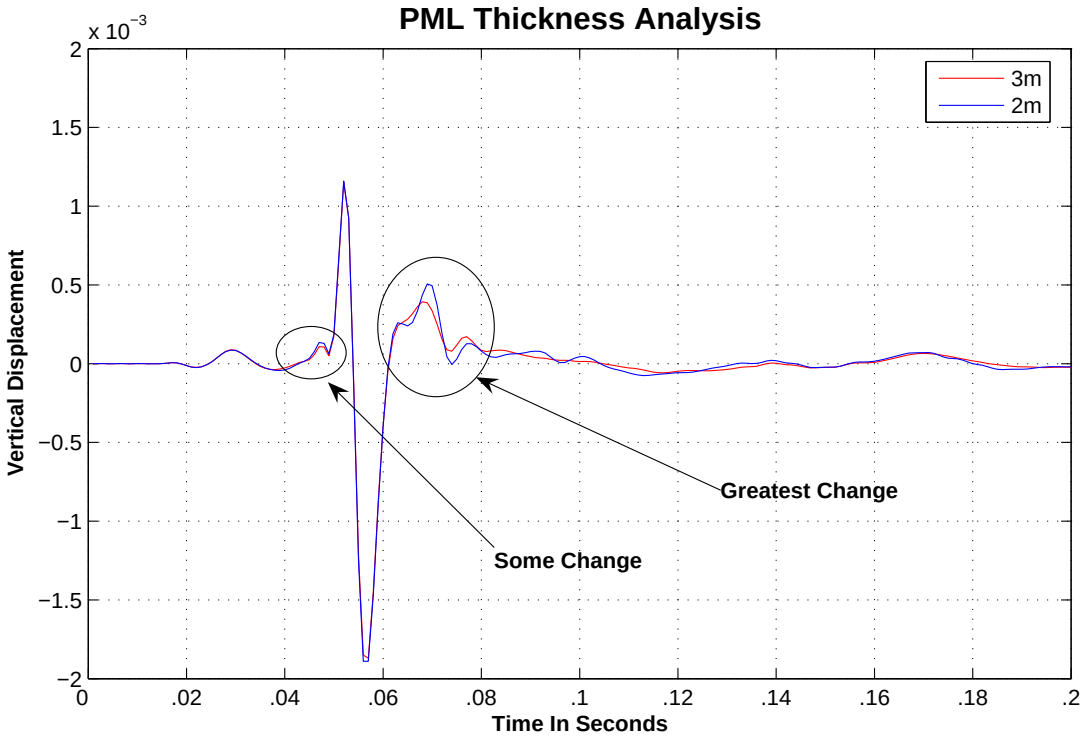


Figure 33. **SAFE-T PML Thickness Analysis (Vertical Displacements) of two PMLs one 3m (i.e., 3 wavelengths) in depth and the other 2m (i.e., 2 wavelengths) in depth. What is remarkable in this analysis is how well a PML as thin as 2m compares to those that are 3m and beyond. Note that the areas of greatest change are not at the peaks and valleys of the Rayleigh wave. This is a huge savings computationally, and it is more efficient. Table III illustrates this fact.**

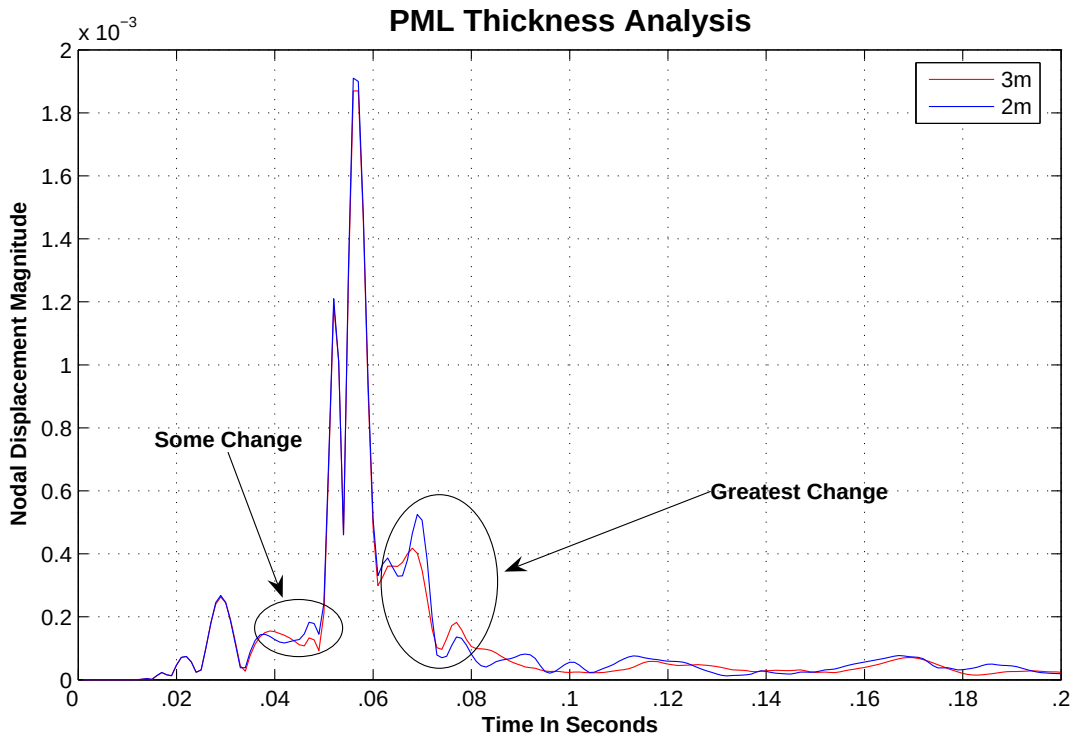


Figure 34. **SAFE-T PML Thickness Analysis (Total Nodal Displacement Magnitudes). As expected, the areas of greatest change are not at the peaks of the total Rayleigh wave disturbance. This makes the PML especially useful in analyzing Rayleigh waves because the error incurred by thinning the PML region accumulates away from the area of interest, that being the magnitude of the Rayleigh wave.**

PML Depth Table					
Depth	**Elements**	**Dofs**	**Equations**	**Memory**	**Time**
5 meters	12800	26082	78246	147.6MB	846.17 s
4 meters	10368	21170	63510	126.0MB	664.16 s
3 meters	8192	16770	50310	98.9MB	505.81 s
2 meters	6272	12882	38646	79.4MB	385.14 s
1 meters	4608	9506	28518	134.9MB	246.24 s

Table II. **This PML Depth Table displays the resource cost of varying the PML thickness.**

PML Efficiency Table					
Depth	**Elements**	**Dofs**	**Equations**	**Memory**	**Time**
5 meters	12800	26082	78246	147.6MB	846.17 s
4 meters	19.0%	18.8%	18.8%	14.6%	21.5%
3 meters	20.9%	20.7%	20.8%	21.5%	23.8%
2 meters	23.4%	23.2%	23.2%	19.7%	23.8%
1 meters	26.5%	26.2%	26.2%	-69.9%	36.1%

Table III. **This table calculates the benefit of reducing PML thickness.**

IV. MODEL END-FIRE ARRAY OF SOURCE THUMPERS

A. INTRODUCTION

A seismic sonar array is a set of n ground source elements distributed over an area of the Earth's surface at a spacing that is selected to allow phasing of the excitation of individual elements to constructively or destructively contribute to a particular source radiation pattern [Ref. 46]. Modeling of a linear end-fire sonar array follows the principles used in classical antenna configuration theory. The total field of an array is a vector superposition of the fields radiated by evenly spaced individual elements. Usually array elements are identical. Directivity can be achieved by *tuning* the array based upon its geometric configuration, distance between elements, amplitude and phase excitations, and the radiating patterns of the individual array elements. Research on seismic SONAR was initiated at the Naval Postgraduate School (NPS) by Dr. Thomas Muir while on leave from the Applied Research Laboratory of the University of Texas at Austin (ARL:UT) in the early to mid 1990s with the goal of determining whether buried mines could be detected in sand. He continued this work at the Naval Postgraduate School when he became chair of the Mine Warfare Department in the late 1990s. [Ref. 47]

Prior research began with Lieutenant(USN) William Stewart. He was first to conduct research related to Seismo-Acoustic SONAR in 1995 [Ref. 48]. He mounted a plunger-type source using a loudspeaker above the ground, and tested the transmitted signal over a wide range of frequencies. His tests were conducted in an above ground swimming pool filled with sand. He was able to show that his source could generate a suitable seismic signal, but the tank was too small for any echo ranging experiments.

In 1998, Lieutenant(USN) Frederick Gaghan [Ref. 49] focussed his research on the development of a discrete-mode excitation source that consisted of two inertial

mass shakers mounted on an aluminum framework. Each inertial mass shaker was mounted to point downward at a 45^o angle in an effort to better excite elliptical Rayleigh surface waves. While promising as an idea, his method needed a more efficient Rayleigh wave source. Lieutenant(USN) Sean Fitzpatrick [Ref. 50] continued Gaghan's work by improving on the source using a linear magnetic force actuator. With a two element seismometer array, he was able to locate 71-291 kg targets at ranges up to 5 meters away. Later that same year, student Major(USMC) Patrick Hall [Ref. 51] measured the reflectivity of targets as a function of their mass load, and found that target reflected signal strength was proportional to target mass.

Captain(USA) Kraig Sheetz [Ref. 52] continued seismic SONAR work in 2000 by developing a receiver that was capable of detecting specific objects such as an M-19, 20 lb, anti-tank mine. Lieutenant(USN) Scott McClelland [Ref. 53] followed Sheetz in 2002 by mounting two inertial shakers onto a manually-pushed rolling cylinder. His source experiments resulted in the successful detection of a 1000-lb bomb at 5 meters. Unfortunately, the roller could only take measurements when the shakers were directly aligned with the ground, and thus it proved less than ideal.

More recently, in 2003, Lieutenant(USN) Douglas MacLean [Ref. 47] introduced a small tracked vehicle with dual inertial mass shakers mounted on top as a mobile source. It excited Rayleigh waves, but additionally generated unwanted P-waves that destructively interfered with signal reception of surface pulses, thereby making the apparatus incapable of finding targets. To mitigate the destructive interference of the P-waves, Lieutenant(USN) Steven E. Rumph [Ref. 9] developed a four-element end-fire array as a seismo-acoustic SONAR capable of being spaced and timed in such a way as to constructively interfere Rayleigh surface waves while simultaneously destructively interfering unwanted P-waves and body waves. Testing on a local beach yielded 3.5 meter beam patterns with approximately 15 db suppression to the rear of the array relative to its forward direction.

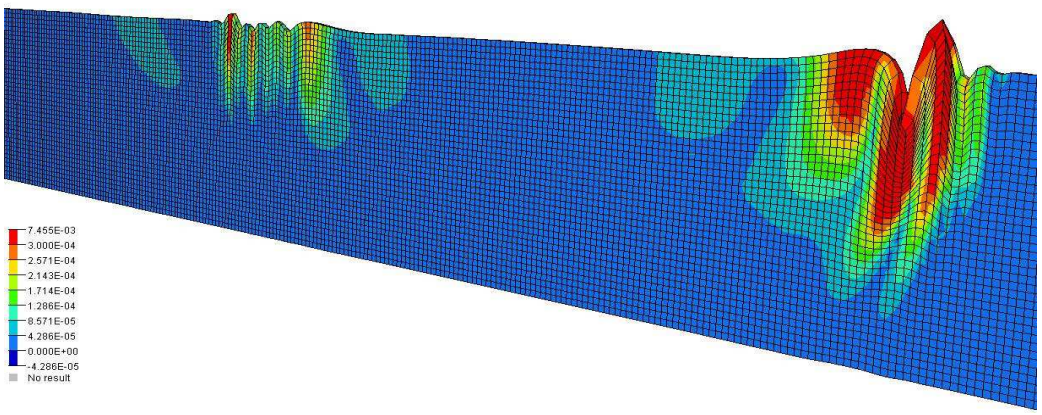


Figure 35. **SAFE-T element by element deformation for medium properties which exhibits a compressional wave speed of 1600 meters per second, a shear wave speed of 100 meters per second, and a Rayleigh wave speed of 95 meters per second.**

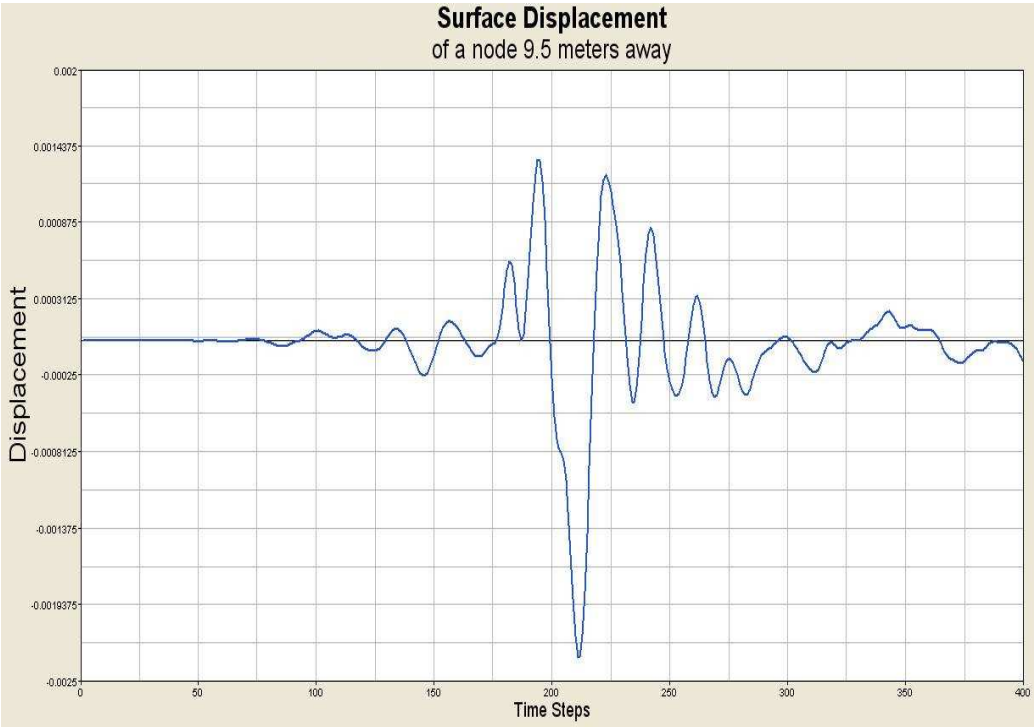


Figure 36. **SAFE-T vertical displacement results with no PML present. Time is represented in Δt time-steps which are 0.0005 seconds apart. The medium is sand. It exhibits a compressional wave speed of 1600 meters per second, a shear wave speed of 100 meters per second, and a Rayleigh wave speed of 95 meters per second.**

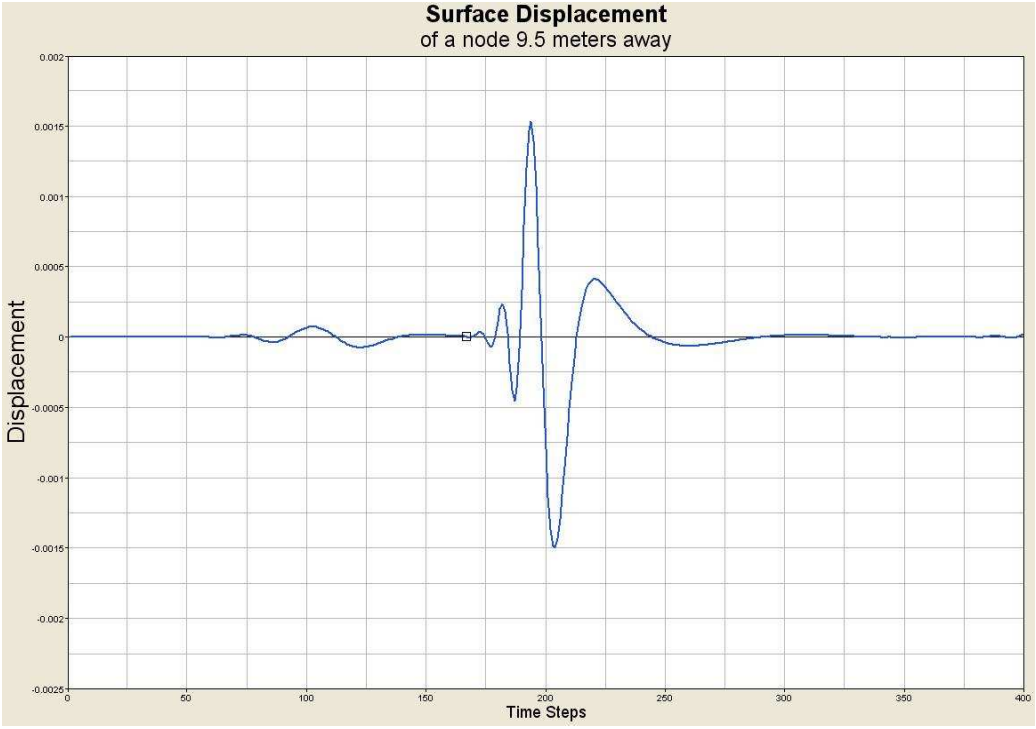


Figure 37. **SAFE-T vertical displacement results with PML present. Notice the smooth transition to rest on the surface after the wave passes. Time is represented in Δt time-steps which are 0.0005 seconds apart. The medium is the same as in figure 36 which exhibits a compressional wave speed of 1600 meters per second, a shear wave speed of 100 meters per second, and a Rayleigh wave speed of 95 meters per second.**

B. SAFE-T RESULTS: OPTIMIZING THE AMPLITUDE OF THE SURFACE WAVE

Numerical results for SAFE-T are presented for a four-element array of downward $(-z)$ source thumpers on a half-space. Figure 35 is an element by element deformation graph postprocessed by Altair Engineering Inc.'s Hyperview v7.0 using SAFE-T's results for a medium with properties that induces a compressional wave speed of 1600 meters per second, a shear wave speed of 100 meters per second, and a Rayleigh wave speed of 95 meters per second inside the computational domain. Figure 36 is a slice of the numerical domain. Lateral edges have rigid Dirichlet boundaries,

and the slice is taken in the $x - z$ plane of the domain. In the first instance, there is no PML present to absorb outgoing waves, thus unwanted body waves bounce back and forth between the rigid boundaries. The material properties of the domain are those closely related to sand, namely, Poisson's ration $\nu = 0.498$, Young's Modulus $E = 8.06E07\ Pa$, and density $\rho = 2690.00\ kg/m^3$. The mesh is very dense (8 elements per meter) in order to provide enough nodes to minimize dispersion of the source pulse using Δt time steps. Displacements are stably computed with fourth order accuracy using equation II.110, 0.0005 seconds apart. The medium exhibits a compressional wave speed of 1600 meters per second, a shear wave speed of 100 meters per second, and a Rayleigh wave speed of 95 meters per second. An arbitrary Gaussian source is used to initiate a four-element end-fire array demonstrating the effects of body waves in a medium without an absorbing layer. Maximum radiation occurs in the direction along the line of the array designated as the positive x direction equivalent to 0° for all calculations. SAFE-T demonstrates its ability to effectively absorb unwanted body waves from the surface of the computational domain in figure 37. The attenuation component of the damping function, equation III.16, is chosen to be linear in the PML. The vertical displacement results show a smooth transition to rest on the surface after the wave passes. Further, because of the speed and relatively small amplitude of the compressional wave, the only waves clearly visible in the graph are the shear and Rayleigh waves.

Figure 38 shows the response of a node which models a particles motion on the free surface of the half-space 5 meters from the end of the end-fire array. It is directly in the path of maximum radiation, i.e., when $\theta = 0^\circ$ or along the axis of the array. The total field of the four-element array is a vector superposition of the disturbances generated by the individual element thumpers. In order to provide a more directive pattern, it is necessary to have the partial fields (generated by the individual thumpers) interfere constructively in the direction of maximum excitation and interfere destructively in the remaining wave propagating space.

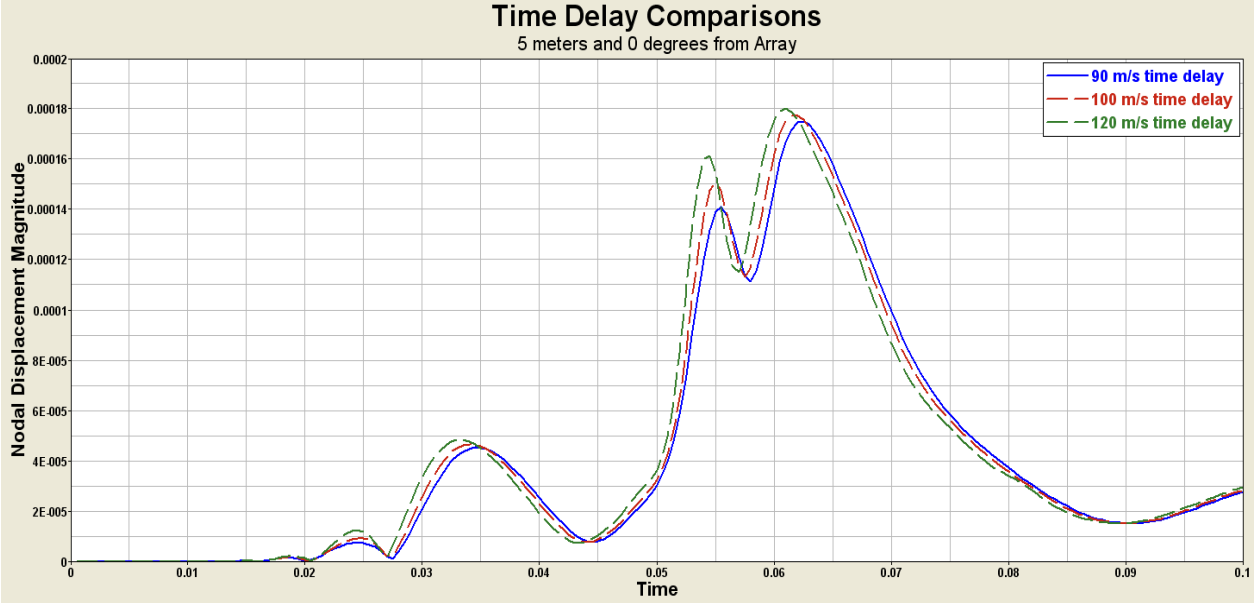


Figure 38. **This graph shows the response of a node located on the free surface and 5 meters away from the End-fire array. It is directly in the path of maximum radiation, i.e., when $\theta = 0^\circ$ along the axis of the array.**

a. Time Delay and Optimized End-fire Array

Of the five generally excepted methods used to control array patterns, i.e., geometrical configuration, displacement between thumpers, excitation amplitude of each thumper, excitation phase of each thumper, relative pattern of each thumper, the methods found most effective in this FE model were space between thumpers and phase. The phase is controlled through the use of time delay. Time delay is accomplished through the relation

$$Time\ Delay = \frac{Distance\ Between\ Elements}{Wave\ Speed}, \tag{IV.1}$$

and provides an effective means to steer through interference an array with a finite number of seismic elements.

Time Delay Table			
Array Characteristics	**Distance Apart** in meters	**Wave Speed** in meters per second	**Time Delay** in seconds
4el Linear	0.25	90 m/s	0.002778 s
4el Linear	0.25	95 m/s	0.002631 s
4el Linear	0.25	100 m/s	0.002500 s
4el Linear	0.25	120 m/s	0.002083 s
4el Linear	0.25	130 m/s	0.001923 s
4el Linear	0.25	140 m/s	0.001785 s

Table IV. **Time delay conversions for various wave speeds allow the end-fire array to be optimized for maximum radiation along the axis of propagation.**

Table IV gives time delay conversions for select wave speeds. Figure 39 shows the effects of destructive interference due to time delay. This destructive interference occurs when $\theta = 180°$ along the axis of the array which corresponds to minimal surface wave radiation. There is a marked difference in the amount of destructive interference depending on the time delay used. In this model, among the three delays tested, a time delay of 0.002778 seconds provides the most destructive interference behind the array.

The reaction of particles to body waves traveling underneath the end-fire array is of primary concern. Wood's(1968) [Ref. 30] experiments show that there is a considerable amount of energy traveling down and away from the surface. In order to optimize the energy steered by the array 0° on the surface and along the positive axis of the end-fire array, energy traveling down must be minimized.

Figure 40 depicts the nodal displacement magnitude response of a particle 5 meters underneath the end-fire array and above the PML in a non layered media . The largest suppression of energy occurs when a 0.002778 second time delay

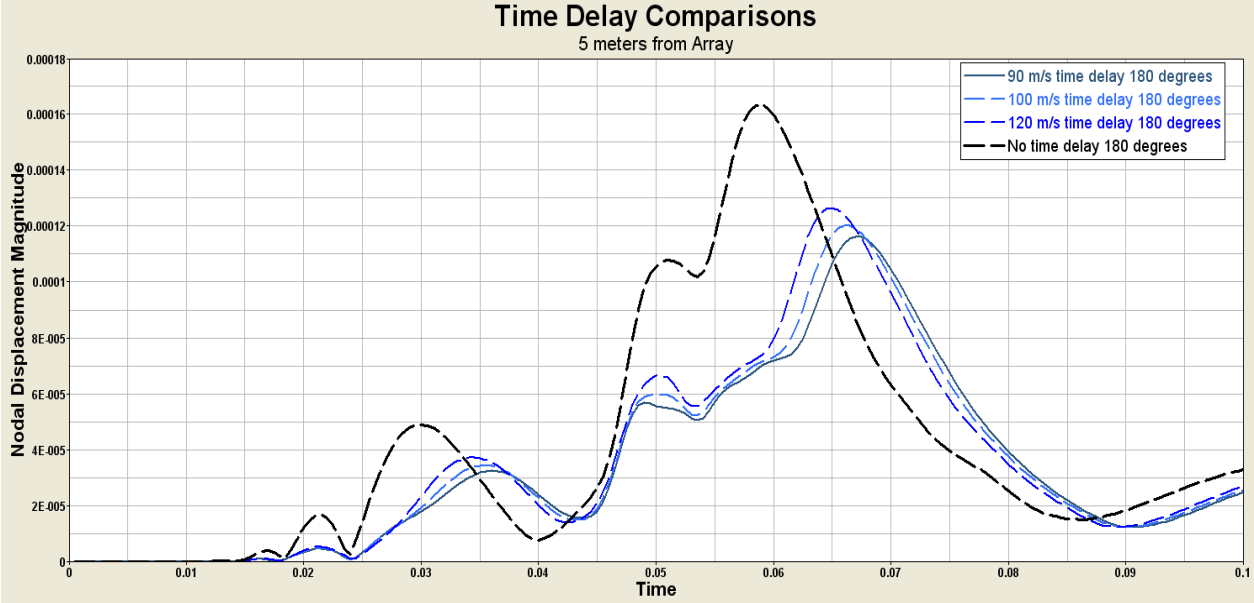


Figure 39. **This graph shows the response of a node located on the free surface and 5 meters away from the End-fire array. It is directly in the path of minimum radiation, i.e., when $\theta = 180°$ along the axis of the array. The effects of time delay clearly shows a dramatic reduction in the amount of energy traveling opposite the direction of steering.**

is applied. This corresponds to a wave speed of 90 meters per second. The other two time delays, 0.0025 seconds (100 meters per second) and 0.002083 seconds (120 meters per second), also show a suppression of body waves when compared to the non time-delayed wave strength. Figure 41 shows the effects of constructive interference as the timing of the excitation of individual elements in the array contribute to boosting the wave's energy. The propagating strength of the wave traveling at 0° and along the positive axis of maximum radiation is higher than the non time-delayed wave. Figure 42 combines into one graph minimal surface nodal magnitude ($\theta = 180°$), maximal surface nodal magnitude ($\theta = 0°$), and the nodal displacement effects of downward body waves. Figure 43 uses the derivative of the Gaussian wave form (figure 12) as an input source to the end-fire array. This source works very well mathematically

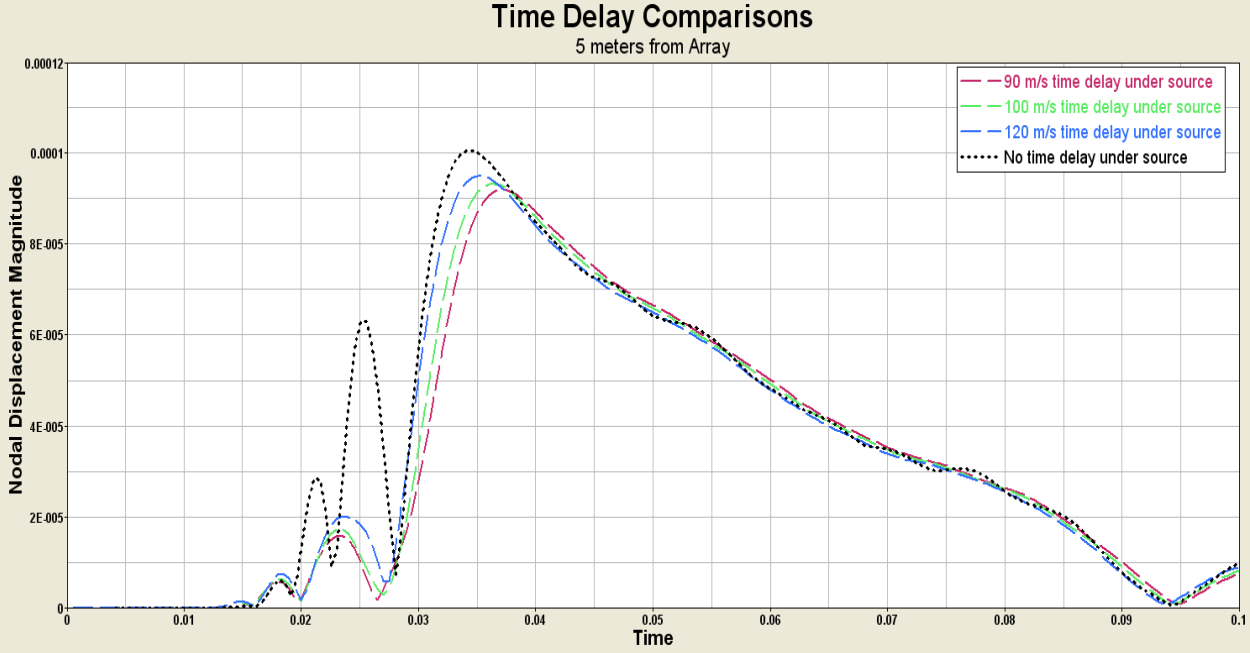


Figure 40. **This graph shows the response of a node located 5 meters under an end-fire array. It is composed of mostly shear and compressional wave components. Though slight, there is a decrease in energy propagating under the array for different time delays.**

because it generates a better Rayleigh wave. By specifically tuning the source to a particular Rayleigh wave speed (see Appendix C), the derivative of the Gaussian best takes advantage of the PML used to truncated the computational domain. Figure 44 are the results of the optimal time delay for achieving the highest gain at 0° and along the positive axis of the array. The delay is 0.002631 seconds which corresponds to a wave speed of 95 meters per second. It corresponds to the minimum radiation at 180° and under the array.

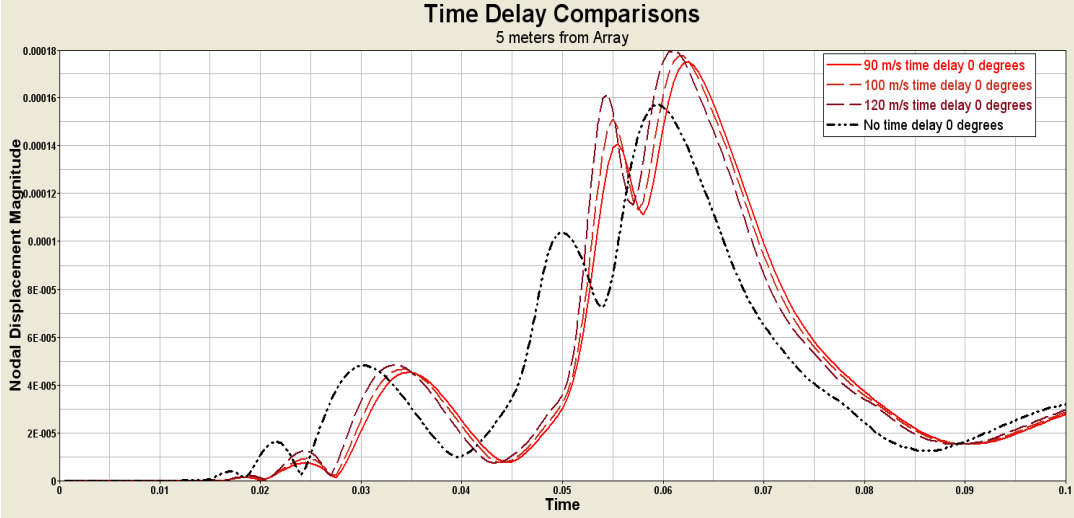


Figure 41. **Time delay comparisons taken at 0 degrees and 5 meters away. It includes the effects of time delay by showing the propagating energy when no time delay is present.**

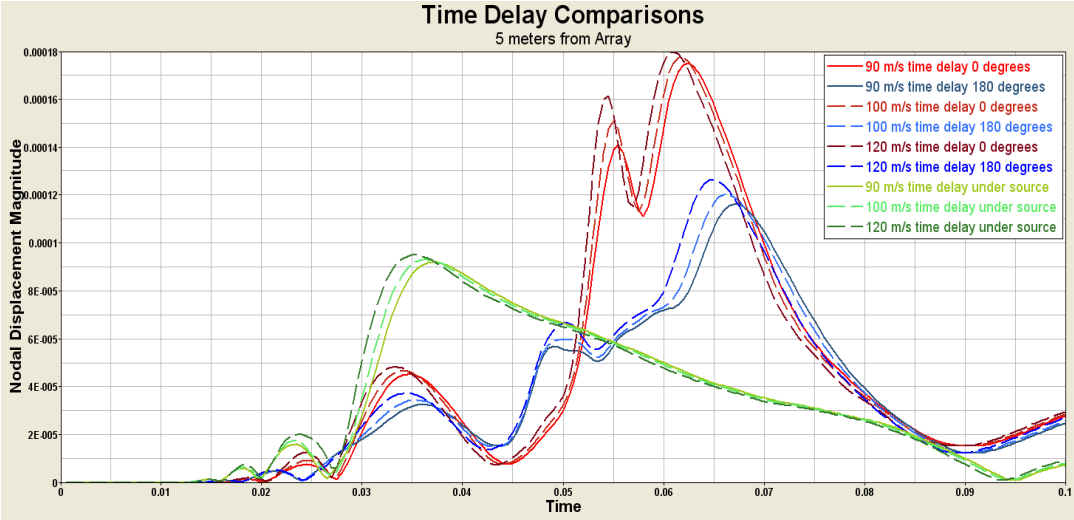


Figure 42. **This graph simultaneously displays time delay results for surface front, surface rear, and sub surface wave propagation nodal displacement magnitudes.**

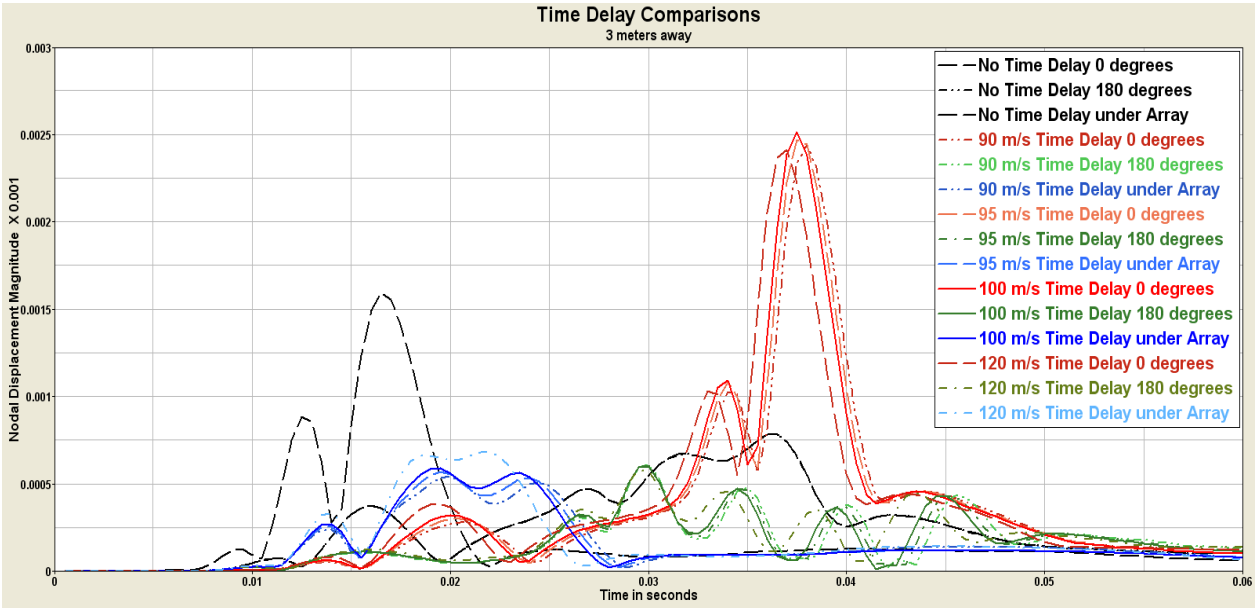


Figure 43. **The use of the derivative of the Gaussian pulse as a time delayed input source to the end-fire array takes best advantage of the PML.**

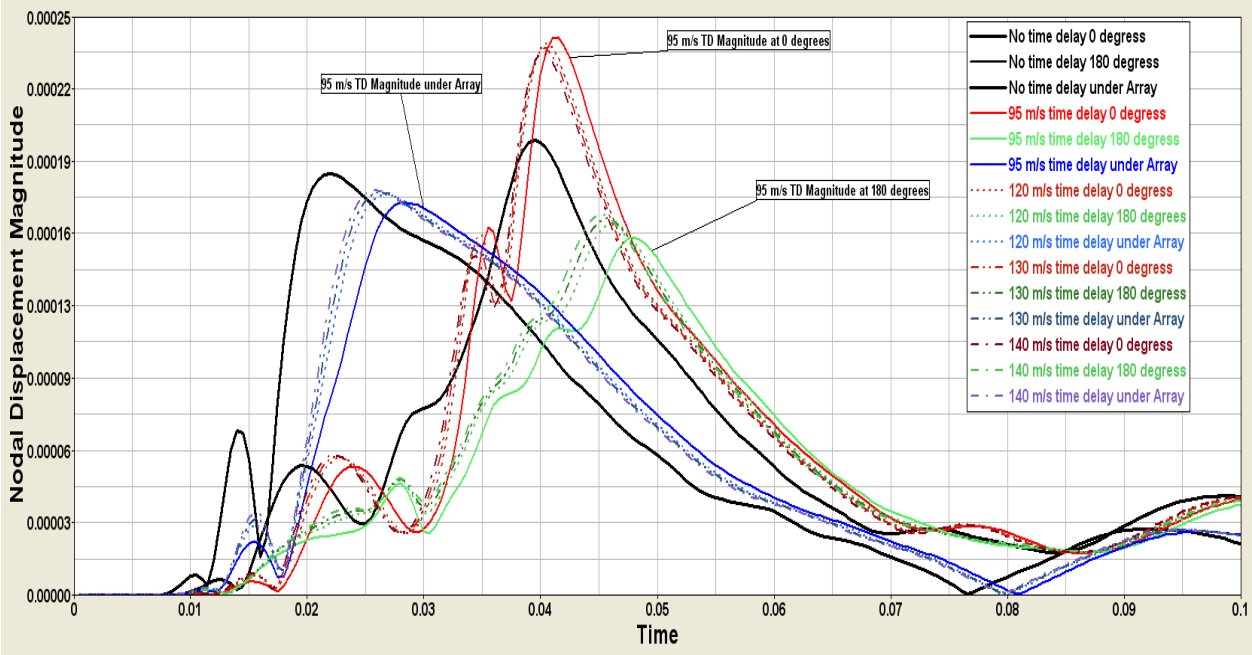


Figure 44. **The optimal time delay for achieving the highest gain at** 0° **and along the positive axis of the array occurs at 0.002631 seconds which corresponds to a wave speed of 95 meters per second. The times for other end-fire time delays for a material with properties,** $\lambda = 683.26E07\ Pa$, $\mu = 2.69E07\ Pa$, $\nu = 0.498$, **Young's Modulus** $E = 8.06E07\ Pa$, **and density** $\rho = 2690.00\ kg/m^3$ **are analyzed and compared.**

V. FINDINGS AND CONCLUSIONS

A solid time dependent perfectly matched layer has been developed to absorb propagating waves which result from a seismic event. This dissertation presents the major steps involved in building a transient PML model for an isotropic, homogeneous media applied to truncating the computational domain where elements of an end-fire array are excited. The following summarizes the findings and conclusions of this dissertation.

• Determined method to find suitable analytic benchmarks for seismic wave analysis.

In order to measure the accuracy of any finite element code, suitable benchmarks have to be analytically computed. For the seismic pulse on a half-space, analytic computations involved an instantaneous pressure that produces a singularity that propagates as the surface wave. Numerically, this presents a significant challenge. How does one determine the magnitude? As the singular point is approached, the amplitude of the disturbance approaches infinity. As a result of this dissertation, finite analytic solutions exist that allow numerical methods to be verified for accuracy and efficiency.

• Development of a new strain-stress equation which included both the perfectly matched media and the computational domain.

This dissertation presented the first known three dimensional, vector-valued, time dependent stress-strain relation from which the strain and stress of a three dimensional solid system within a perfectly matched medium was calculated. This strain equation gives the damping media inhomogeneous elastic properties that attenuated propagating waves in the PML region. Because the damping properties are dependent upon the location of the wave, all effects of the attenuation vanish inside the computational domain.

• Development of 3D time dependent perfectly matched layer.

A new unsplit 3D time dependent perfectly matched layer was derived and converted into its weak (Galerkin) form for implementation into SAFE-T (Solid Adaptive Finite Element Transient), developed by the author for seismic wave analysis. This dissertation demonstrated SAFE-T's ability to accurately model seismic phenomena using perfectly matched layers to absorb unwanted reflected surface and body waves.

• Determined that linearity of damping function did not contribute greatly to damping region properties.

The damping functions used within the PML region determine the speed at which propagating waves are attenuated. A comparison of the effects of using linear, quadratic, and cubic damping functions showed that each provided excellent absorption. The analysis did not reveal a significant difference in the rate of attenuation.

• Determined that the damping amplitude when made too great has the effect of stiffening the PML/computational domain interface causing reflections.

The damping constant stiffened the interface by essentially making what should be a gradual increase in attenuation a more abrupt change thereby causing an unwanted reflection. The analysis of this dissertation demonstrates the need to choose a damping constant that provides maximum amount of attenuation with the least amount of stiffness at an interface.

• Determined that reflections cost considerable computer resources when undamped.

Table II and table III demonstrate an unintended consequences of not damping surface and body waves, namely,CPU memory. This is not intuitive, but since every motion in the FE model need be calculated, it stands to reason that unwanted reflections would expend computer resources. In fact, the analysis reveals that although a 1 meter PML requires less time (a mere 246.24 seconds to compute-compared to 846.17 seconds for the 5 meter PML), it is a disaster for computer memory. It consumes an inordinate amount of computer memory. It requires 91.4% of the memory that would be needed for a model with a PML 5 meters deep. The dissertation demonstrates

further the need to attenuate unwanted reflections within a FE model.

• Determined the optimal spacing and timing for maximal Rayleigh displacement magnitude and minimal body wave magnitudes given known material properties.

SAFE-T calculated the optimal time delay and space between elements for achieving the highest Rayleigh surface wave gain along the axis of the end-fire array. This positive axis equates to 0^o and corresponds to the end of the array that produces the largest Rayleigh wave. Concurrently, the analysis was consistent with array theory, i.e., as the magnitude of the Rayleigh wave increased, SAFE-T clearly showed that the amplitude of the body waves decreased.

As a result of the developments involved in this investigation, several future research opportunities exist. Those efforts should include, but not limit themselves to the following:

• Investigating methods to make the PML dynamic, i.e., include logic into the mc.ff (MCOEFF) FORTRAN routine to sense wave motion and calculate wave speed for the damping function.

• Place obstacles into the computational domain and perform scattering calculations and source level estimations on a variety of array configurations.

• Analysis of non-homogeneous/anisotropic materials.

• Analysis of non-linear wave phenomenon such as shock waves. The PML can be tuned to attenuate non-linear waves as well.

• Conduct a time dependent analysis of an infinite waveguide using the transient PML as an infinite boundary.

• Examine the usefulness of method in non-destructive testing of elastic members of mechanical devices such as aircraft wings, nuclear cooling pipes, and ship hull analysis.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. A (STRESS TERMS)

1. UNKNOWN STRESS TERMS

$$\int_{\Omega} \mathbf{S}^{A}_{jm} \left(\lambda \mathbf{S}^{B}_{kk} \mathbf{F}_{k} u^{n+1}_{kk} \delta_{ij} + \mu \mathbf{S}^{B}_{ij} \left[\mathbf{F}_{i} u^{n+1}_{i,j} + \mathbf{F}_{j} u^{n+1}_{j,i} \right] \right) v_{i,m} \partial\Omega \tag{.1}$$

a. Main Diagonal

For the main diagonal the terms will be as follows:

$$\tau^{n+1}_{11} v_{1,1} = \mathbf{S}^{A}_{11} \left[(\lambda + 2\mu) \, \mathbf{S}^{B}_{11} \mathbf{F}_{1} \; u^{n+1}_{1,1} + \lambda \mathbf{S}^{B}_{22} \mathbf{F}_{2} \; u^{n+1}_{2,2} + \lambda \mathbf{S}^{B}_{33} \mathbf{F}_{3} \; u^{n+1}_{3,3} \right] \tag{.2}$$

$$\tau^{n+1}_{22} v_{2,2} = \mathbf{S}^{A}_{22} \left[\lambda \mathbf{S}^{B}_{11} \mathbf{F}_{1} \; u^{n+1}_{1,1} + (\lambda + 2\mu) \, \mathbf{S}^{B}_{22} \mathbf{F}_{2} \; u^{n+1}_{2,2} + \lambda \mathbf{S}^{B}_{33} \mathbf{F}_{3} \; u^{n+1}_{3,3} \right] \tag{.3}$$

$$\tau^{n+1}_{33} v_{3,3} = \mathbf{S}^{A}_{33} \left[\lambda \mathbf{S}^{B}_{11} \mathbf{F}_{1} \; u^{n+1}_{1,1} + \lambda \mathbf{S}^{B}_{22} \mathbf{F}_{2} \; u^{n+1}_{2,2} + (\lambda + 2\mu) \, \mathbf{S}^{B}_{33} \mathbf{F}_{3} \; u^{n+1}_{3,3} \right] \tag{.4}$$

b. Cross Terms

The cross terms will be as follows:

$$\tau^{n+1}_{12} v_{1,2} = \mathbf{S}^{A}_{22} \mu \left(\mathbf{S}^{B}_{12} \mathbf{F}_{1} \; u^{n+1}_{1,2} + \mathbf{S}^{B}_{21} \mathbf{F}_{2} \; u^{n+1}_{2,1} \right) \tag{.5}$$

$$\tau^{n+1}_{13} v_{1,3} = \mathbf{S}^{A}_{33} \mu \left(\mathbf{S}^{B}_{13} \mathbf{F}_{1} \; u^{n+1}_{1,3} + \mathbf{S}^{B}_{31} \mathbf{F}_{3} \; u^{n+1}_{3,1} \right) \tag{.6}$$

$$\tau^{n+1}_{21} v_{2,1} = \mathbf{S}^{A}_{11} \mu \left(\mathbf{S}^{B}_{21} \mathbf{F}_{2} \; u^{n+1}_{2,1} + \mathbf{S}^{B}_{12} \mathbf{F}_{1} \; u^{n+1}_{1,2} \right) \tag{.7}$$

$$\tau^{n+1}_{23} v_{2,3} = \mathbf{S}^{A}_{33} \mu \left(\mathbf{S}^{B}_{23} \mathbf{F}_{2} \; u^{n+1}_{2,3} + \mathbf{S}^{B}_{32} \mathbf{F}_{3} \; u^{n+1}_{3,2} \right) \tag{.8}$$

$$\tau^{n+1}_{31} v_{3,1} = \mathbf{S}^{A}_{11} \mu \left(\mathbf{S}^{B}_{31} \mathbf{F}_{3} \; u^{n+1}_{3,1} + \mathbf{S}^{B}_{13} \mathbf{F}_{1} \; u^{n+1}_{1,3} \right) \tag{.9}$$

$$\tau^{n+1}_{32} v_{3,2} = \mathbf{S}^{A}_{22} \mu \left(\mathbf{S}^{B}_{32} \mathbf{F}_{3} \; u^{n+1}_{3,2} + \mathbf{S}^{B}_{23} \mathbf{F}_{2} \; u^{n+1}_{2,3} \right) \tag{.10}$$

2. INITIAL AND KNOWN STRESS TERMS

$$- \int_{\Omega} \left(\mathbf{S}^{A}_{jm} \chi_{ij} + \mathbf{A}_{jm} \Delta t \sum_{l=1}^{n} \tau^{l}_{ij} + \tilde{\mathbf{A}}_{jm} \Delta t^{2} \sum_{l=1}^{n} ((n+1) - l) \tau^{l}_{ij} \right) v_{i,m} \partial\Omega \tag{.11}$$

a. Main Diagonal

For the main diagonal the terms will be as follows:

$$v_{1,1} = -\mathbf{S}^{A}_{11} \chi_{11} - \mathbf{A}_{11} \Delta t \sum_{l=1}^{n} \tau^{l}_{11} - \tilde{\mathbf{A}}_{11} \Delta t^{2} \sum_{l=1}^{n} ((n+1) - l) \tau^{l}_{11} \tag{.12}$$

$$v_{2,2} = -\mathbf{S}^A_{22}\chi_{22} - \mathbf{A}_{22}\Delta t \sum_{l=1}^{n} \tau^l_{22} - \tilde{\mathbf{A}}_{22}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau^l_{22} \tag{.13}$$

$$v_{3,3} = -\mathbf{S}^A_{33}\chi_{33} - \mathbf{A}_{33}\Delta t \sum_{l=1}^{n} \tau^l_{33} - \tilde{\mathbf{A}}_{33}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau^l_{33} \tag{.14}$$

b. Cross Terms

The cross terms will be as follows:

$$v_{1,2} = -\mathbf{S}^A_{22}\chi_{12} - \mathbf{A}_{22}\Delta t \sum_{l=1}^{n} \tau^l_{12} - \tilde{\mathbf{A}}_{22}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau^l_{12} \tag{.15}$$

$$v_{1,3} = -\mathbf{S}^A_{33}\chi_{13} - \mathbf{A}_{33}\Delta t \sum_{l=1}^{n} \tau^l_{13} - \tilde{\mathbf{A}}_{33}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau^l_{13} \tag{.16}$$

$$v_{2,1} = -\mathbf{S}^A_{11}\chi_{21} - \mathbf{A}_{11}\Delta t \sum_{l=1}^{n} \tau^l_{21} - \tilde{\mathbf{A}}_{11}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau^l_{21} \tag{.17}$$

$$v_{2,3} = -\mathbf{S}^A_{33}\chi_{23} - \mathbf{A}_{33}\Delta t \sum_{l=1}^{n} \tau^l_{23} - \tilde{\mathbf{A}}_{33}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau^l_{23} \tag{.18}$$

$$v_{3,1} = -\mathbf{S}^A_{11}\chi_{31} - \mathbf{A}_{11}\Delta t \sum_{l=1}^{n} \tau^l_{31} - \tilde{\mathbf{A}}_{11}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau^l_{31} \tag{.19}$$

$$v_{3,2} = -\mathbf{S}^A_{22}\chi_{32} - \mathbf{A}_{22}\Delta t \sum_{l=1}^{n} \tau^l_{32} - \tilde{\mathbf{A}}_{22}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau^l_{32} \tag{.20}$$

3. INITIAL AND KNOWN BOUNDARY TERMS

$$\int_{\Gamma_t} \hat{t}^n_i(x,y,z) v_i ds \tag{.21}$$

a. Three Components

$$v_1 = \hat{t}^n_1(x,y,z) \tag{.22}$$

$$v_2 = \hat{t}^n_2(x,y,z) \tag{.23}$$

$$v_3 = \hat{t}^n_3(x,y,z) \tag{.24}$$

APPENDIX. B (MCOEFF TERMS)

1. MCOEFF TERMS

Subroutine MCOEFF is called for every integration point and evaluates all of the components of the coefficient matrices for a single element or a batch of elements. The A_{ij} PHLEX coefficients are identified from the final Galerkin (weak) form of the PDE.

a. LHS Coefficients

Body forces induced as time marches.

$$(u_1^{n+1}v_1)A_{00}(1,1) = \kappa_1 + f_k + \frac{\Delta t f_l}{2} \tag{.25}$$

$$(u_2^{n+1}v_2)A_{00}(2,2) = \kappa_1 + f_k + \frac{\Delta t f_l}{2} \tag{.26}$$

$$(u_3^{n+1}v_3)A_{00}(3,3) = \kappa_1 + f_k + \frac{\Delta t f_l}{2} \tag{.27}$$

Set 1

$$(u_{1,1}^{n+1}v_{1,1})A_{11}(1,1) = (\lambda + 2\mu)\,\mathbf{S}_{11}^A\mathbf{S}_{11}^B\mathbf{F}_1 \tag{.28}$$

$$(u_{1,2}^{n+1}v_{1,2})A_{22}(1,1) = \mu\mathbf{S}_{22}^A\mathbf{S}_{12}^B\mathbf{F}_1 \tag{.29}$$

$$(u_{2,2}^{n+1}v_{1,1})A_{12}(1,2) = \lambda\mathbf{S}_{11}^A\mathbf{S}_{11}^B\mathbf{F}_1 \tag{.30}$$

$$(u_{2,1}^{n+1}v_{1,2})A_{21}(1,2) = \mu\mathbf{S}_{22}^A\mathbf{S}_{21}^B\mathbf{F}_2 \tag{.31}$$

$$(u_{1,3}^{n+1}v_{1,3})A_{33}(1,1) = \mu\mathbf{S}_{33}^A\mathbf{S}_{13}^B\mathbf{F}_1 \tag{.32}$$

$$(u_{3,3}^{n+1}v_{1,1})A_{13}(1,3) = \lambda\mathbf{S}_{11}^A\mathbf{S}_{11}^B\mathbf{F}_1 \tag{.33}$$

$$(u_{3,1}^{n+1}v_{1,3})A_{31}(1,3) = \mu\mathbf{S}_{33}^A\mathbf{S}_{31}^B\mathbf{F}_3 \tag{.34}$$

Set 2

$$(u_{2,1}^{n+1}v_{2,1})A_{11}(2,2) = \mu\mathbf{S}_{11}^A\mathbf{S}_{21}^B\mathbf{F}_2 \tag{.35}$$

$$(u_{2,2}^{n+1}v_{2,2})A_{22}(2,2) = (\lambda + 2\mu)\,\mathbf{S}_{22}^A\mathbf{S}_{22}^B\mathbf{F}_2 \tag{.36}$$

$$(u_{1,2}^{n+1}v_{2,1})A_{12}(2,1) = \mu\mathbf{S}_{11}^A\mathbf{S}_{12}^B\mathbf{F}_1 \tag{.37}$$

$$(u_{1,1}^{n+1}v_{2,2})A_{21}(2,1) = \lambda \mathbf{S}_{22}^{A}\mathbf{S}_{22}^{B}\mathbf{F}_2 \tag{.38}$$

$$(u_{2,3}^{n+1}v_{2,3})A_{33}(2,2) = \mu \mathbf{S}_{33}^{A}\mathbf{S}_{23}^{B}\mathbf{F}_2 \tag{.39}$$

$$(u_{3,3}^{n+1}v_{2,2})A_{23}(2,3) = \lambda \mathbf{S}_{22}^{A}\mathbf{S}_{22}^{B}\mathbf{F}_2 \tag{.40}$$

$$(u_{3,2}^{n+1}v_{2,3})A_{32}(2,3) = \mu \mathbf{S}_{33}^{A}\mathbf{S}_{32}^{B}\mathbf{F}_3 \tag{.41}$$

Set 3

$$(u_{1,3}^{n+1}v_{3,1})A_{13}(3,1) = \mu \mathbf{S}_{11}^{A}\mathbf{S}_{13}^{B}\mathbf{F}_1 \tag{.42}$$

$$(u_{1,1}^{n+1}v_{3,3})A_{31}(3,1) = \lambda \mathbf{S}_{33}^{A}\mathbf{S}_{33}^{B}\mathbf{F}_3 \tag{.43}$$

$$(u_{2,3}^{n+1}v_{3,2})A_{23}(3,2) = \mu \mathbf{S}_{22}^{A}\mathbf{S}_{23}^{B}\mathbf{F}_2 \tag{.44}$$

$$(u_{2,2}^{n+1}v_{3,3})A_{32}(3,2) = \lambda \mathbf{S}_{33}^{A}\mathbf{S}_{33}^{B}\mathbf{F}_3 \tag{.45}$$

$$(u_{3,1}^{n+1}v_{3,1})A_{11}(3,3) = \mu \mathbf{S}_{11}^{A}\mathbf{S}_{31}^{B}\mathbf{F}_3 \tag{.46}$$

$$(u_{3,2}^{n+1}v_{3,2})A_{22}(3,3) = \mu \mathbf{S}_{22}^{A}\mathbf{S}_{32}^{B}\mathbf{F}_3 \tag{.47}$$

$$(u_{3,3}^{n+1}v_{3,3})A_{33}(3,3) = (\lambda + 2\mu)\, \mathbf{S}_{33}^{A}\mathbf{S}_{33}^{B}\mathbf{F}_3 \tag{.48}$$

b. RHS Coefficients

$$(v_1)f(1) = \kappa_1 u_1^n + \kappa_2 \frac{\partial u_1^n}{\partial t} + \kappa_3 \frac{\partial^2 u_1^n}{\partial t^2} - f_l \Delta t \sum_{l=1}^{n} u_1^l \tag{.49}$$

$$(v_2)f(2) = \kappa_1 u_2^n + \kappa_2 \frac{\partial u_2^n}{\partial t} + \kappa_3 \frac{\partial^2 u_2^n}{\partial t^2} - f_l \Delta t \sum_{l=1}^{n} u_2^l \tag{.50}$$

$$(v_3)f(3) = \kappa_1 u_3^n + \kappa_2 \frac{\partial u_3^n}{\partial t} + \kappa_3 \frac{\partial^2 u_3^n}{\partial t^2} - f_l \Delta t \sum_{l=1}^{n} u_3^l \tag{.51}$$

$$(v_{1,1})f_x(1) = -\mathbf{S}_{11}^{A}\chi_{11} - \mathbf{A}_{11}\Delta t \sum_{l=1}^{n} \tau_{11}^{l} - \tilde{\mathbf{A}}_{11}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau_{11}^{l} \tag{.52}$$

$$(v_{1,2})f_x(2) = -\mathbf{S}_{22}^{A}\chi_{12} - \mathbf{A}_{22}\Delta t \sum_{l=1}^{n} \tau_{12}^{l} - \tilde{\mathbf{A}}_{22}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau_{12}^{l} \tag{.53}$$

$$(v_{1,3})f_x(3) = -\mathbf{S}_{33}^{A}\chi_{13} - \mathbf{A}_{33}\Delta t \sum_{l=1}^{n} \tau_{13}^{l} - \tilde{\mathbf{A}}_{33}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau_{13}^{l} \tag{.54}$$

$$(v_{2,1})f_y(1) = -\mathbf{S}_{11}^{A}\chi_{21} - \mathbf{A}_{11}\Delta t \sum_{l=1}^{n} \tau_{21}^{l} - \tilde{\mathbf{A}}_{11}\Delta t^2 \sum_{l=1}^{n}((n+1)-l)\tau_{21}^{l} \tag{.55}$$

$$(v_{2,2})f_y(2) = -\mathbf{S}^A_{22}\chi_{22} - \mathbf{A}_{22}\Delta t\sum_{l=1}^{n}\tau^l_{22} - \tilde{\mathbf{A}}_{22}\Delta t^2\sum_{l=1}^{n}((n+1)-l)\tau^l_{22} \qquad (.56)$$

$$(v_{2,3})f_y(3) = -\mathbf{S}^A_{33}\chi_{23} - \mathbf{A}_{33}\Delta t\sum_{l=1}^{n}\tau^l_{23} - \tilde{\mathbf{A}}_{33}\Delta t^2\sum_{l=1}^{n}((n+1)-l)\tau^l_{23} \qquad (.57)$$

$$(v_{3,1})f_z(1) = -\mathbf{S}^A_{11}\chi_{31} - \mathbf{A}_{11}\Delta t\sum_{l=1}^{n}\tau^l_{31} - \tilde{\mathbf{A}}_{11}\Delta t^2\sum_{l=1}^{n}((n+1)-l)\tau^l_{31} \qquad (.58)$$

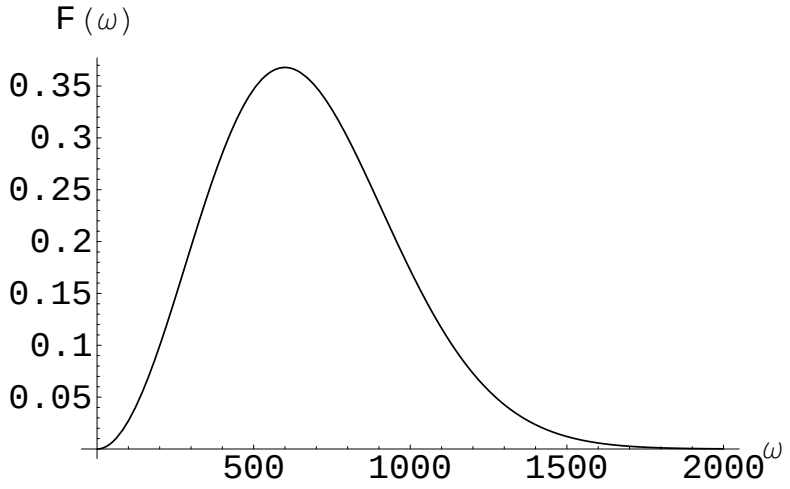
$$(v_{3,2})f_z(2) = -\mathbf{S}^A_{22}\chi_{32} - \mathbf{A}_{22}\Delta t\sum_{l=1}^{n}\tau^l_{32} - \tilde{\mathbf{A}}_{22}\Delta t^2\sum_{l=1}^{n}((n+1)-l)\tau^l_{32} \qquad (.59)$$

$$(v_{3,3})f_z(3) = -\mathbf{S}^A_{33}\chi_{33} - \mathbf{A}_{33}\Delta t\sum_{l=1}^{n}\tau^l_{33} - \tilde{\mathbf{A}}_{33}\Delta t^2\sum_{l=1}^{n}((n+1)-l)\tau^l_{33} \qquad (.60)$$

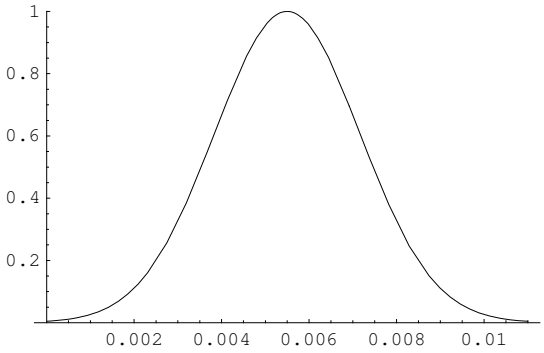
THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. C (SOURCE COMPUTATIONS)

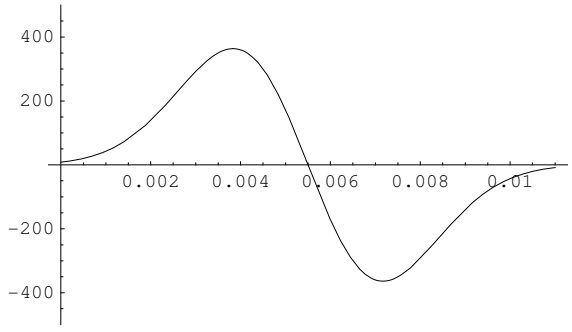
The following computer code found in this appendix presents the procedures used to construct the source used for the seismic sonar array. It was crafted in such a way as to produce a Rayleigh wave of unit wavelength for material properties which match closely with that of sand. It is written and evaluated using Wolfram's Mathematica 5.2.



```
In[1]:= Off[FindMaximum::"lstol", General::"spell1"]

In[2]:= Amplitude = 1;
        Peak = 0.005500;
        Wid = 0.0023583;
        f[t_] := Amplitude e^-(t-Peak)^2/Wid^2
        f[t]
        f'[t]
        ftplot = Plot[f[t], {t, 0, 2 Peak}, PlotRange → {0, Amplitude}]
        derftplot = Plot[f'[τ], {τ, 0, 2 Peak}, PlotRange → {-500.0, 500.0}]

Out[6]= e^-179805. (-0.0055+t)^2

Out[7]= -359610. e^-179805. (-0.0055+t)^2 (-0.0055 + t)
```




```
Out[8]= - Graphics -
```




```
Out[9]= - Graphics -

In[10]:= FindRoot[f'[t] == 0, {t, 0.001, 0.006}]

Out[10]= {t → 0.0055}

In[11]:= FindRoot[f''[t] == 0, {t, 0, 0.004}]

Out[11]= {t → 0.00383243}
```

```
In[12]:= anst = t /. %[[1]]
```

```
Out[12]= 0.00383243
```

```
In[13]:= f'[anst]
```

```
Out[13]= 363.721
```

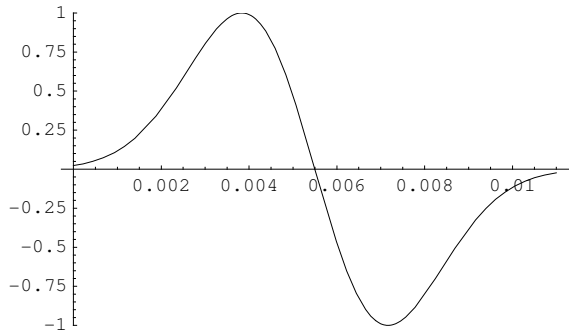
```
In[14]:= Solve[h f'[anst] == 1, h]
```

```
Out[14]= {{h → 0.00274936}}
```

```
In[15]:= ansh = h /. %[ 1]]
```

```
Out[15]= 0.00274936
```

```
In[16]:= derftplot = Plot[ansh f'[τ], {τ, 0, 2 Peak}, PlotRange → {-1, 1}]
```




```
Out[16]= - Graphics -
```

```
In[17]:= ansh f'[τ]
```

```
Out[17]= -988.697 e^(-179805. (-0.0055+τ)^2) (-0.0055 + τ)
```

```
In[18]:= utrans[ω_] := FourierTransform[f'[t], t, ω]
```

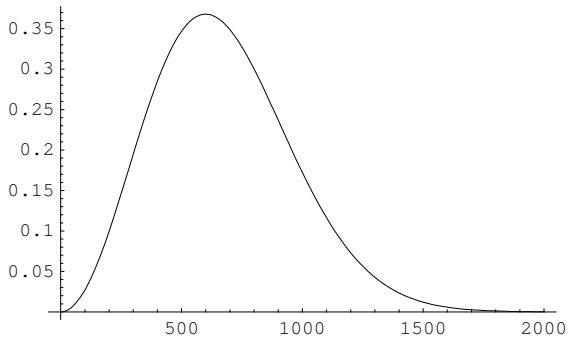
```
In[19]:= utrans[ω]
```

```
Out[19]= -143464.
          (0. e^(-8.67362×10^-19 i ω+0. i ω^2) + e^(1.39039×10^-6 (1977.86+i ω)^2) (2.94319×10^-23 + (0. + 5.04859×10^-11 i) ω) +
            e^(1.39039×10^-6 (1977.86+i ω)^2) ((-4.99269×10^-8 Erf[-2.33219 - 0.00117915 i ω] -
                  4.99269×10^-8 Erf[2.33219 + 0.00117915 i ω]) Sign[1977.86 + i ω]^2 +
               ((-4.99269×10^-8 + 0. i) - (0. + 2.52429×10^-11 i) ω)
                (Erf[-2.33219 - (0. + 0.00117915 i) ω] + Erf[2.33219 + (0. + 0.00117915 i) ω])
                Sign[2.33219 + (0. + 0.00117915 i) ω]^2))
```

```
In[20]:= Abs[utrans[1]]^2 / N
```

```
Out[20]= 2.78078×10^-6
```

```
In[21]:= Plot[Abs[utrans[ω]]^2, {ω, 0, 2000}, PlotRange → Automatic, PlotPoints → 100]
```




```
Out[21]= - Graphics -
```

```
In[22]:= FindMaximum[Abs[utrans[ω] , {ω, 500}]
```

```
Out[22]= {0.606531, {ω → 599.675}}
```

```
In[23]:= ω = 599.6749964809572`
```

```
Out[23]= 599.675
```

Looking for a wavelength of 1 meter. Use $\frac{\omega}{2\pi} = f$ and then $\frac{\omega}{2\pi}$ = wavelength $* c_R$

```
In[24]:= ω / (2 π)
```

```
Out[24]= 95.4412
```

Material properties for sand...

```
In[25]:= ρ = 2690 kg / m^3
         μ = 2.69 * 10^7 kg / (s^2 m);
         λ = 683.26 * 10^7 kg / (s^2 m);
```

```
Out[25]= 2690 kg / m^3
```

Determine Poisson's Ratio given the material properties...

```
In[28]:= ν = λ / (2 (λ + μ))
```

```
Out[28]= 0.498039
```

```
In[29]:= cs = PowerExpand[(μ/ρ)^(1/2)]
```

$$\text{Out[29]=} \quad \frac{100.\ \text{m}}{\text{s}}$$

Computing the Rayleigh wave speed using Achenbach...

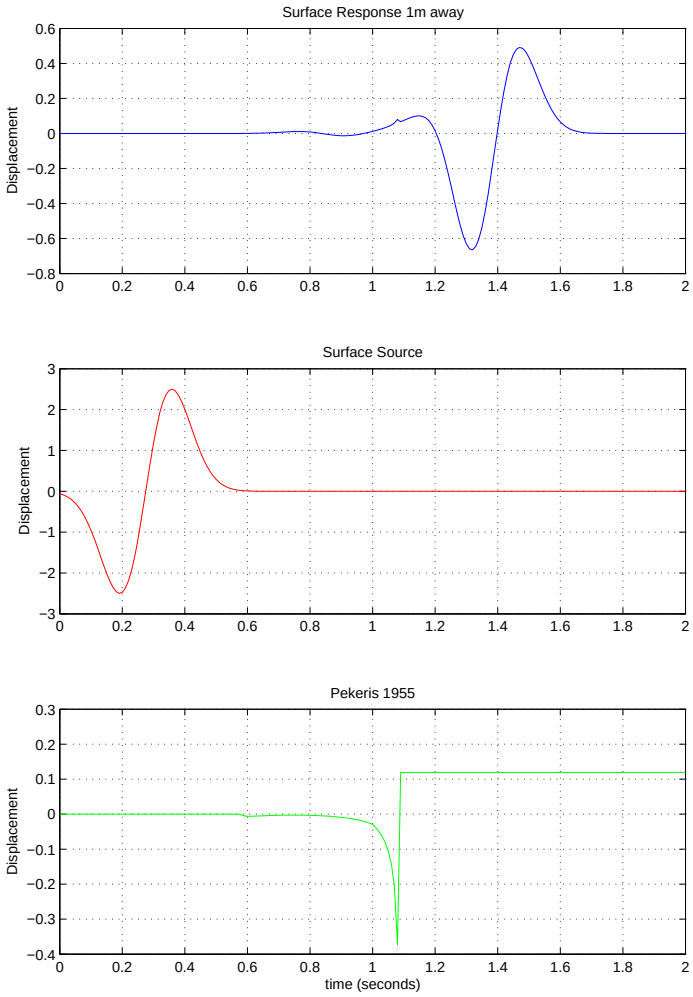
```
In[30]:= cR = (.862 + 1.14 ν)/(1 + ν) cs
```

$$\text{Out[30]=} \quad \frac{95.4424\ \text{m}}{\text{s}}$$

THIS PAGE INTENTIONALLY LEFT BLANK

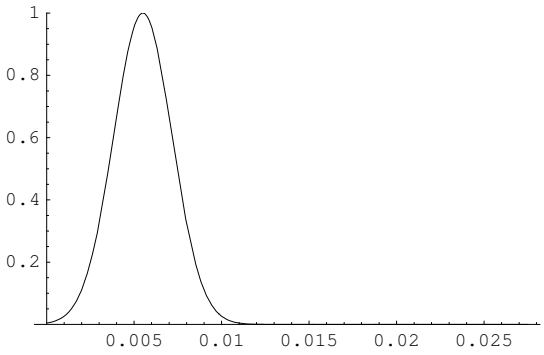
APPENDIX. D (ARRAY SUPERPOSITION CALCULATIONS)

This appendix gives the Mathematica code used to examine the analytic properties of an end-fire array.

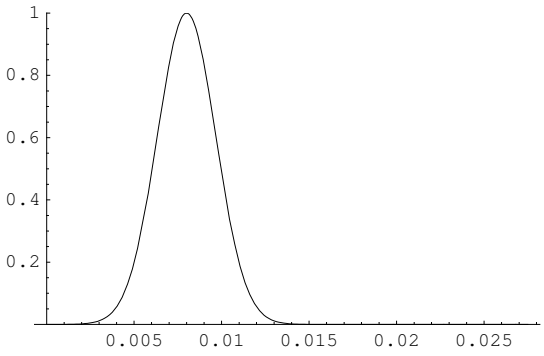


Endfire 1/4-lambda Array w/ Gaussian Source

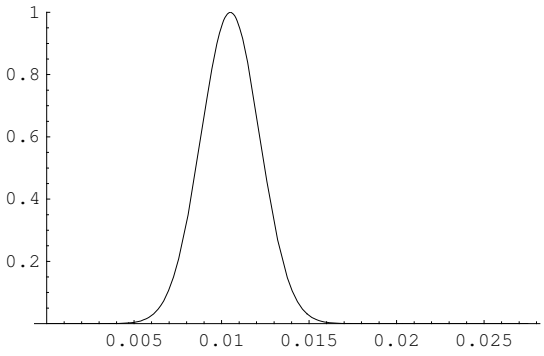
```
In[1]:= Amplitude = 1;
        Peak1 = .0055;
        Wid1 = .0023583;
        Peak2 = .0080;
        Wid2 = .0023583;
        Peak3 = .0105;
        Wid3 = .0023583;
        Peak4 = .0130;
        Wid4 = .0023583;
        f1[t_] := Amplitude e^-(t-Peak1)^2/Wid1^2
        f2[t_] := Amplitude e^-(t-Peak2)^2/Wid2^2
        f3[t_] := Amplitude e^-(t-Peak3)^2/Wid3^2
        f4[t_] := Amplitude e^-(t-Peak4)^2/Wid4^2
        ftplot1 = Plot[f1[t], {t, 0, 5 Peak1}, PlotRange → {0, Amplitude}]
        ftplot2 = Plot[f2[t], {t, 0, 5 Peak1}, PlotRange → {0, Amplitude}]
        ftplot3 = Plot[f3[t], {t, 0, 5 Peak1}, PlotRange → {0, Amplitude}]
        ftplot4 = Plot[f4[t], {t, 0, 5 Peak1}, PlotRange → {0, Amplitude}]
```



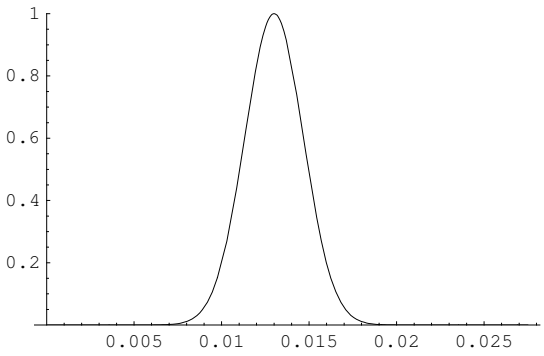

```
Out[14]= - Graphics -
```



Out[15]= **-** Graphics **-**

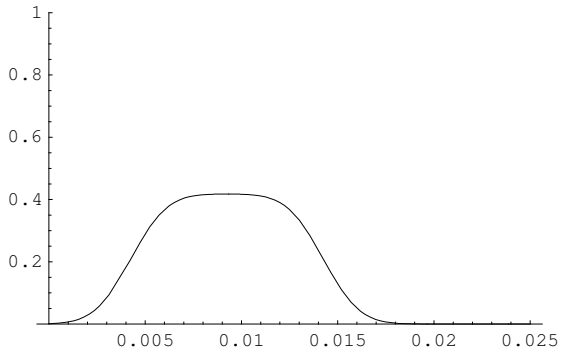


Out[16]= **-** Graphics **-**



Out[17]= **-** Graphics **-**

In[18]:=
```
g1[t_] := (f1[t] + f2[t] + f3[t] + f4[t]) / 4
Plot[g1[t], {t, 0, 0.025}, PlotRange → {0, Amplitude}]
```



Out[19]= - Graphics -

In[20]:=
```
utrans[ω_] := FourierTransform[g1[t], t, ω]
```

```
utrans[ω]
```

$$\frac{1}{4\sqrt{2\pi}}\left(0.0000181552\, e^{1.39039\times10^{-6}\,(1977.86+i\,\omega)^2} + 9.07761\times10^{-6}\, e^{1.39039\times10^{-6}\,(1977.86+i\,\omega)^2}\right.$$
$$(1.\,\mathrm{Erf}[-2.33219 - 0.00117915\,i\,\omega] + 1.\,\mathrm{Erf}[2.33219 + 0.00117915\,i\,\omega]$$
$$\mathrm{Sign}[1977.86 + i\,\omega]^2 + 2.10131\times10^{-8}\, e^{1.39039\times10^{-6}\,(2876.88+i\,\omega)^2}$$
$$(2. + (1.\,\mathrm{Erf}[-3.39227 - 0.00117915\,i\,\omega] + 1.\,\mathrm{Erf}[3.39227 + 0.00117915\,i\,\omega])$$
$$\mathrm{Sign}[2876.88 + i\,\omega]^2) + 5.1393\times10^{-12}\, e^{1.39039\times10^{-6}\,(3775.91+i\,\omega)^2}$$
$$(2. + (1.\,\mathrm{Erf}[-4.45236 - 0.00117915\,i\,\omega] + 1.\,\mathrm{Erf}[4.45236 + 0.00117915\,i\,\omega])$$
$$\mathrm{Sign}[3775.91 + i\,\omega]^2) + 1.32805\times10^{-16}\, e^{1.39039\times10^{-6}\,(4674.93+i\,\omega)^2}$$
$$(2. + (1.\,\mathrm{Erf}[-5.51245 - 0.00117915\,i\,\omega] + 1.\,\mathrm{Erf}[5.51245 + 0.00117915\,i\,\omega])$$
$$\left.\mathrm{Sign}[4674.93 + i\,\omega]^2)\right)$$

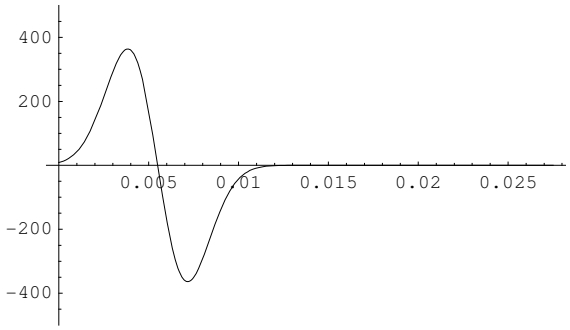
In[21]:=
```
Abs[utrans[.01]]^2 / N
```

Out[21]= 1.08932×10^{-6}

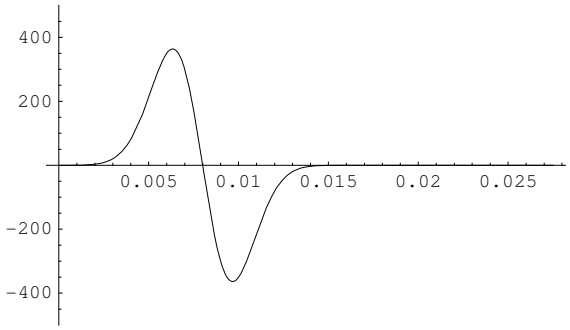
In[22]:=
```
Plot[Abs[utrans[ω]]^2, {ω, 0, 2000}, PlotRange → Automatic, PlotPoints → 100]
```

Endfire 1/4-lambda Array w/ Derivative of Gaussian Source

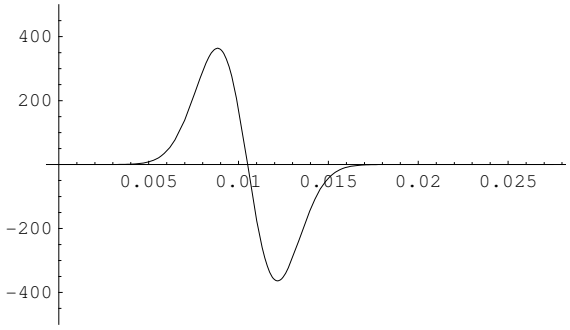
```
In[1]:= Amplitude = 1;
        Peak1 = .0055;
        Wid1 = .0023583;
        Peak2 = .0080;
        Wid2 = .0023583;
        Peak3 = .0105;
        Wid3 = .0023583;
        Peak4 = .0130;
        Wid4 = .0023583;
        f1[t_] := Amplitude e^-(t-Peak1)^2/Wid1^2
        f2[t_] := Amplitude e^-(t-Peak2)^2/Wid2^2
        f3[t_] := Amplitude e^-(t-Peak3)^2/Wid3^2
        f4[t_] := Amplitude e^-(t-Peak4)^2/Wid4^2
        ftplot1 = Plot[f1'[t], {t, 0, 5 Peak1}, PlotRange → {-500 Amplitude, 500 Amplitude}]
        ftplot2 = Plot[f2'[t], {t, 0, 5 Peak1}, PlotRange → {-500 Amplitude, 500 Amplitude}]
        ftplot3 = Plot[f3'[t], {t, 0, 5 Peak1}, PlotRange → {-500 Amplitude, 500 Amplitude}]
        ftplot4 = Plot[f4'[t], {t, 0, 5 Peak1}, PlotRange → {-500 Amplitude, 500 Amplitude}]
```



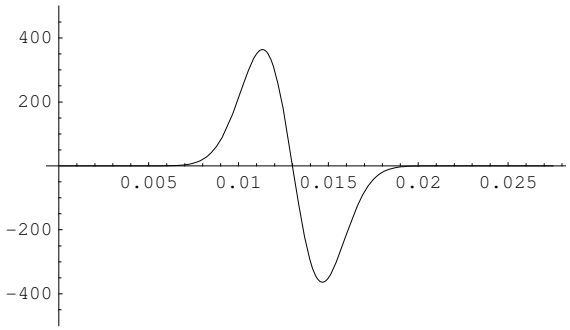

```
Out[14]= - Graphics -
```



Out[15]= **-** Graphics **-**

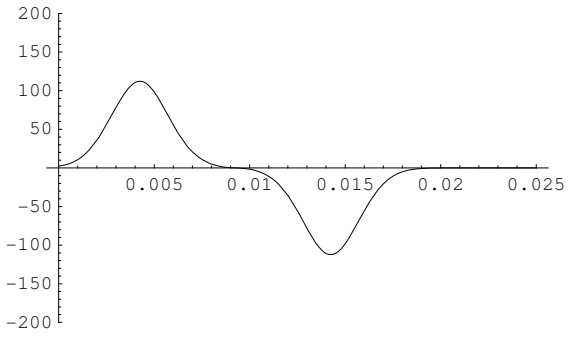


Out[16]= **-** Graphics **-**



Out[17]= **-** Graphics **-**

In[18]:=
```
g2[t_] := (f1'[t] + f2'[t] + f3'[t] + f4'[t]) / 4
Plot[g2[t], {t, 0, 0.025}, PlotRange → {-200 Amplitude, 200 Amplitude}]
```



Out[19]= - Graphics -

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. E (PROPHLEX FORTRAN MODULE)

```
c  --- mc.ff ---- Fri Jul 26 14:29:13 CDT 2006
c   Copyright: Computational Mechanics, Co., Inc. 1992-1996
c   MAJ Anthony N. Johnson, Naval Postgraduate School
c   Monterey, California 93940
c=============================================================
 #include "PH-stdF.h"
#include "PH-parameters.h"
#include "PH-Fmacros.h"
#include "application-parameters.h"
#include "application-macros.h"
c
c============================================================ mc =======
c
      subroutine mc(el2get,
     &     a00,a01,a02,a03,
     &     a10,a11,a12,a13,
     &     a20,a21,a22,a23,
     &     a30,a31,a32,a33,
     &     f,fx,fy,fz,
     &     xyz, xyznod, ngnode,
     &     dxdxi,dxidx,xjac,
     &     numel,ncomp,nca,nsol,
     &     u,uxyz,
     &     xi,eta,zeta,bcinfo)
c
c**********************************************************************
c*                                                                    *
c*  Routine: mc                                                       *
c*  Purpose: define the interior integral coefficients of the         *
c*           variational problem.                                     *
c*  Variables:                                                        *
c*  ----------                                                        *
c*  I  el2get:        element number to load if > 0                   *
c*                      otherwise batch flag if <= 0                  *
c*  O  a00,...,a33:   coefficients of the contribution to the stiffness*
c*                      matrix from the interior integral coded as:   *
c*                      apq:  (p,q) = {0,1,2,3} where:                *
c*                        0:  signifies the shape function itself     *
c*                        1:  signifies x-derivative of shape function *
c*                        2:  signifies y-derivative of shape function *
c*                        3:  signifies z-derivative of shape function *
c*                        p:  first index of apq signifies test func.  *
c*                        q:  second index of apq signifies trial func-*
c*                              tion (or the solution and its derivs.) *
c*  O  f,...,fz:      coefficients of the contribution to the load    *
c*                      vector of the interior integral coded as:     *
c*                      fp: p = { ,n,t,s} where:                      *
c*                         :  signifies shape function itself         *
c*                        x:  signifies x-derivative of shape function *
c*                        y:  signifies y-derivative of shape function *
c*                        z:  signifies z-derivative of shape function *
c*                        p:  signifies a function multiplied by the  *
c*                              test function and its derivatives.    *
c*  I  xyz:           integratin point coords                         *
c*  I  xyznod:        nodal point coordinates for each element        *
c*  I  ngnode:        number of nodes per element                     *
c*  I  dxdxi:         dx/dxi                                          *
c*  I  dxidx:         dxi/dx                                          *
```

```
c*  I  xjac           jacobians for each element                      *
c*  I  numel:         number of elements in the batch                 *
c*  I  ncomp:         number of solution components                   *
c*  I  nca:           number of active solution components            *
c*  I  nsol:          number of solutions                             *
c*  I  u,uxyz:        solution and its derivatives at the intgr. point *
c*  I  xi,eta,zeta    integration point coords in master coords       *
c*  P  bcinfo         boundary condition info                         *
c*                                                                    *
c********************************************************************
c
#include "PH-std.blk"
#include "PH-ftnwrk.blk"
#include "PH-wrkspcPtrs.blk"
c
      REALTYPE
     &     a00,a01,a02,a03,
     &     a10,a11,a12,a13,
     &     a20,a21,a22,a23,
     &     a30,a31,a32,a33,
     &     f,fx,fy,fz,
     &     xyz, xyznod,
     &     dxdxi,dxidx,xjac,
     &     u,uxyz,xi,eta,zeta
c
      REALTYPE xlambdat, xmut, xrhot, pi
      REALTYPE beta, gamma, deltat
      REALTYPE fprp, fevn,
     &         pmlfx, pmlfy, pmlfz, cs
  REALTYPE fm,fc,fk,fl,k1,k2,k3,
     &         tc1,tc2,tc3,tc4,tc5,tc6
      REALTYPE sum_e, sum_uij, sum_u,
     &         sum_T, sum2_T, sum2_e,
     &         etn, etnp1, Ecur, Tcur
      REALTYPE Lpmlx, Lpmly, Lpmlz
      REALTYPE mx, my, mz, bx, by, bz
      REALTYPE pmlstartx, pmlstarty, pmlstartz
c
      INTTYPE el2get, ngnode, numel, nod
      INTTYPE ncomp, nca, nsol, lastep1
      INTTYPE soltn, soltnp1, veltn, acctn
      INTTYPE k, l, lastep2
c
      CPTRTYPE elptr,bcinfo
c
      INTTYPE ielstrt, ielend, curstep
      INTTYPE icomp, jcomp, iel
      INTTYPE debug, pml_on
      INTTYPE passnum, iresid, idiag
c
c
      dimension
     &     a00(numel,ncomp,ncomp),
     &     a01(numel,ncomp,ncomp),
     &     a02(numel,ncomp,ncomp),
     &     a03(numel,ncomp,ncomp),
     &     a10(numel,ncomp,ncomp),
     &     a11(numel,ncomp,ncomp),
     &     a12(numel,ncomp,ncomp),
     &     a13(numel,ncomp,ncomp),
     &     a20(numel,ncomp,ncomp),
     &     a21(numel,ncomp,ncomp),
     &     a22(numel,ncomp,ncomp),
     &     a23(numel,ncomp,ncomp),
```

```
     &     a30(numel,ncomp,ncomp),
     &     a31(numel,ncomp,ncomp),
     &     a32(numel,ncomp,ncomp),
     &     a33(numel,ncomp,ncomp)

      dimension
     &     etn(MAXELEMBATCH,ncomp,ncomp),
     &     etnp1(MAXELEMBATCH,ncomp,ncomp),
     &     Ecur(MAXELEMBATCH,ncomp,ncomp),
     &     Tcur(MAXELEMBATCH,ncomp,ncomp),
     &     sum_e(MAXELEMBATCH,ncomp,ncomp),
     &     sum2_e(MAXELEMBATCH,ncomp,ncomp),
     &     sum_T(MAXELEMBATCH,ncomp,ncomp),
     &     sum_uij(MAXELEMBATCH,ncomp,ncomp),
     &     sum2_T(MAXELEMBATCH,ncomp,ncomp)
      dimension
     &     u(numel,ncomp,nsol),
     &     sum_u(MAXELEMBATCH,ncomp,nsol),
     &     uxyz(numel,ncomp,nsol,3),
     &     f(numel,ncomp),
     &     fprp(MAXELEMBATCH,ncomp),
     &     fevn(MAXELEMBATCH,ncomp),
     &     fx(MAXELEMBATCH,ncomp),
     &     fy(MAXELEMBATCH,ncomp),
     &     fz(MAXELEMBATCH,ncomp)
      dimension
     &     xyz(numel,3), xyznod(numel,ngnode,3),
     &     dxdxi(numel,3,3),dxidx(numel,3,3),xjac(numel),
     &     bcinfo(numel,7)
      dimension
     &     xlambdat(MAXELEMBATCH),
     &     xmut(MAXELEMBATCH),
     &     xrhot(MAXELEMBATCH),
     &     fm(MAXELEMBATCH),
     &     fc(MAXELEMBATCH),
     &     fk(MAXELEMBATCH),
     &     fl(MAXELEMBATCH),
     &     k1(MAXELEMBATCH),
     &     k2(MAXELEMBATCH),
     &     k3(MAXELEMBATCH),
     &     tc1(MAXELEMBATCH),
     &     tc2(MAXELEMBATCH),
     &     tc3(MAXELEMBATCH),
     &     tc4(MAXELEMBATCH),
     &     tc5(MAXELEMBATCH),
     &     tc6(MAXELEMBATCH),
     &     pmlfx(MAXELEMBATCH),
     &     pmlfy(MAXELEMBATCH),
     &     pmlfz(MAXELEMBATCH),
     &     cs(MAXELEMBATCH)
c
       pml_on = 1
c    debug = 1
c-----------------------------------------------------------------------
c load the time step parameter data
c
      call GTAPPROP(beta,gamma)
      call GTPARAM( NONL_PARAMS, DTNONL, deltat)
      call GTPARAM(NONL_PARAMS, ITSTEP, curstep)

c------ set up the parallel batch stuff
c
      call PH_GET_BATCH_( el2get, numel, ielstrt, ielend )
c
```

```
c------ get residual flag, the diagonal preconditioner flag, and
c       the pass number flag
c
      PH_GET_SWITCH_(S_resid, iresid)
      PH_GET_SWITCH_(S_diag, idiag)
      PH_GET_SWITCH_(S_passnum, passnum)
c
c
      call PH_UG_GET_SLOT_(SOLUTION_TN, soltn)
      call PH_UG_GET_SLOT_(VELOCITY_TN, veltn)
      call PH_UG_GET_SLOT_(ACCELERATION_TN, acctn)
c
c load up the material data for the element batch
c
          do 3101 iel = ielstrt,ielend
c
         elptr = PH_GET_ELEMENT_POINTER_FROM_BATCHID_(iel)
         call gethooke ( elptr, xlambdat(iel), xmut(iel), xrhot(iel),
     &             pmlfx(iel), pmlfy(iel), pmlfz(iel) )
c
        if (xrhot(iel) .eq. 0) then
    cs(iel)=0.0d0
else
    cs(iel)=sqrt(xmut(iel)/xrhot(iel))
endif

c
           do 3111 nod = 1, ngnode
c
            if (abs(xyznod(iel,nod,1)) .gt. Lpmlx) then
              Lpmlx = abs(xyznod(iel,nod,1))
c               print *, 'Lpmlx', Lpmlx
c               pmlstartx = Lpmlx
            endif
c
            if (abs(xyznod(iel,nod,2)) .gt. Lpmly) then
              Lpmly = abs(xyznod(iel,nod,2))
c               print *, 'Lpmly', Lpmly
c               pmlstarty = Lpmly
            endif
c
            if (abs(xyznod(iel,nod,3)) .gt. Lpmlz) then
              Lpmlz = abs(xyznod(iel,nod,3))
c               print *, 'Lpmlz', Lpmlz
c               pmlstartz = Lpmlz
            endif
c
3111      continue
3101      continue
c
      do 4001 iel = ielstrt,ielend
c------ calculate propagating wave functions
c
c
         fprp(iel,1) = abs(pmlfx(iel)*(xyznod(iel,2,1)/Lpmlx)**2)
         fprp(iel,2) = abs(pmlfy(iel)*(xyznod(iel,2,2)/Lpmly)**2)
         fprp(iel,3) = abs(pmlfz(iel)*(xyznod(iel,2,3)/Lpmlz)**2)
c
c------ calculate evenescant wave functions
c
         fevn(iel,1) = abs(pmlfx(iel)*(xyznod(iel,2,1)/Lpmlx)**2)
         fevn(iel,2) = abs(pmlfy(iel)*(xyznod(iel,2,2)/Lpmly)**2)
         fevn(iel,3) = abs(pmlfz(iel)*(xyznod(iel,2,3)/Lpmlz)**2)
c
```

```
c
4001   continue
c
c------ initialize coefficients
c       note: do not initialize coefficients which are not active
c             ie. if a13 is not active omit the initialization, this
c             will overwrite memory!!!!
c
      do 5001 iel = ielstrt,ielend
        do 5011 icomp = 1,nca
          do 5021 jcomp = 1,nca
             a00(iel,icomp,jcomp) = 0.0d0
             a11(iel,icomp,jcomp) = 0.0d0
             a12(iel,icomp,jcomp) = 0.0d0
             a13(iel,icomp,jcomp) = 0.0d0
             a21(iel,icomp,jcomp) = 0.0d0
             a22(iel,icomp,jcomp) = 0.0d0
             a23(iel,icomp,jcomp) = 0.0d0
             a31(iel,icomp,jcomp) = 0.0d0
             a32(iel,icomp,jcomp) = 0.0d0
             a33(iel,icomp,jcomp) = 0.0d0
  Ecur(iel,icomp,jcomp) = 0.0d0
  Tcur(iel,icomp,jcomp) = 0.0d0
5021     continue
5011    continue
5001  continue
c
c
      do 6101  iel = ielstrt,ielend

c----- compute necessary coefficient values
c
       fm(iel) = xrhot(iel) * (1.0d0 + fevn(iel,1))
     &           * (1.0d0 + fevn(iel,2)) * (1.0d0 + fevn(iel,3))
       fc(iel) = xrhot(iel) * cs(iel) *
     &    (   fprp(iel,1) * (1.0d0 + fevn(iel,2)) * (1.0d0 + fevn(iel,3))
     &      + fprp(iel,2) * (1.0d0 + fevn(iel,1)) * (1.0d0 + fevn(iel,3))
     &      + fprp(iel,3) * (1.0d0 + fevn(iel,1)) * (1.0d0 + fevn(iel,2))  )
       fk(iel) = xrhot(iel) * cs(iel)**2 *
     &    (   fprp(iel,2) * fprp(iel,3) * (1.0d0 + fevn(iel,1))
     &          + fprp(iel,1) * fprp(iel,3) * (1.0d0 + fevn(iel,2))
     &          + fprp(iel,1) * fprp(iel,2) * (1.0d0 + fevn(iel,3))  )
       fl(iel) = xrhot(iel) * cs(iel)**3 *
     &        fprp(iel,1) * fprp(iel,2)* fprp(iel,3)
c
c------ compute generalized newmark time constants
c
      tc1(iel) = 1.0d0/(beta*deltat**2)
      tc2(iel) = 1.0d0/(beta*deltat)
      tc3(iel) = 1.0d0/(2.0d0*beta) - 1.0d0
      tc4(iel) = gamma/(beta*deltat)
      tc5(iel) = (gamma/beta) - 1.0d0
      tc6(iel) = deltat*((gamma/2.0d0*beta) - 1.0d0)
c
c------ compute kappa constants
c
      k1(iel) = fm(iel)*tc1(iel) + fc(iel)*tc4(iel)
      k2(iel) = fm(iel)*tc2(iel) + fc(iel)*tc5(iel)
      k3(iel) = fm(iel)*tc3(iel) + fc(iel)*tc6(iel)
c
 6101 continue
c
c------ initialize time matrices at first time step only
c
```

```
      if (curstep .eq. 1 .and. abs(curstep-lastep1) .ne. 0) then
c
      do 6301  iel = ielstrt,ielend
       do 6311 icomp = 1,nca
        do 6312 jcomp = 1,nca
  sum_e(iel,icomp,jcomp) = 0.0d0
  sum2_e(iel,icomp,jcomp) = 0.0d0
  sum_T(iel,icomp,jcomp) = 0.0d0
  sum2_T(iel,icomp,jcomp) = 0.0d0
  sum_uij(iel,icomp,jcomp) = 0.0d0
  etn(iel,icomp,jcomp) = 0.0d0
  etnp1(iel,icomp,jcomp) = 0.0d0
         sum_u(iel,icomp,jcomp) = 0.0d0
         lastep1 = curstep
6312    continue
6311   continue
6301  continue
c
      call strain(uxyz, numel, ncomp, nca, nsol, deltat, ielstrt,
     .              ielend, fprp, fevn, cs,
     .              Ecur, sum_e, sum2_e, sum_uij, etn, etnp1, curstep)
c
      else if (abs(curstep-lastep1) .ne. 0) then
c
c------ build strain vector after the first time step
c
      call strain(uxyz, numel, ncomp, nca, nsol, deltat, ielstrt,
     .              ielend, fprp, fevn, cs,
     .              Ecur, sum_e, sum2_e, sum_uij, etn, etnp1, curstep)
c
c------ build stress vector
c
c      do 6401  iel = ielstrt,ielend
c
       Tcur(iel,1,1) = xmut(iel)*Ecur(iel,1,1)
       Tcur(iel,1,2) = xmut(iel)*Ecur(iel,1,2)
       Tcur(iel,1,3) = xmut(iel)*Ecur(iel,1,3)
       Tcur(iel,2,1) = xmut(iel)*Ecur(iel,2,1)
       Tcur(iel,2,2) = xmut(iel)*Ecur(iel,2,2)
       Tcur(iel,2,3) = xmut(iel)*Ecur(iel,2,3)
       Tcur(iel,3,1) = xmut(iel)*Ecur(iel,3,1)
       Tcur(iel,3,2) = xmut(iel)*Ecur(iel,3,2)
       Tcur(iel,3,3) = xmut(iel)*Ecur(iel,3,3)
c
       Tcur(iel,1,1) =  Tcur(iel,1,1) + xlambdat(iel)
     & *(Ecur(iel,1,1)+Ecur(iel,2,2)+Ecur(iel,3,3))
       Tcur(iel,2,2) =  Tcur(iel,2,2) + xlambdat(iel)
     & *(Ecur(iel,1,1)+Ecur(iel,2,2)+Ecur(iel,3,3))
       Tcur(iel,3,3) =  Tcur(iel,3,3) + xlambdat(iel)
     & *(Ecur(iel,1,1)+Ecur(iel,2,2)+Ecur(iel,3,3))
c
c 6401  continue
c
         sum_T = sum_T + Tcur
c
         sum2_T = (curstep+curstep**2)*Tcur
c
         sum_u = sum_u + u
c
         lastep1 = curstep
c
      endif
c
      if(idiag .eq. GETPRECDIAG .or. passnum .eq. GETLHS) then
```

```
c
        do 8101 iel = ielstrt,ielend
c
c------ load up the lhs coefficients
c
c note the coefficients are stored as follows
c      lambda   =  xlambdat(iel)
c      mu       =  xmut(iel)
c      rho      =  xrhot(iel)
c
c
c
       if (pml_on .eq. 1) then
c
c
c
          a00(iel,1,1) =  k1(iel)+fk(iel)+0.5*deltat*fl(iel)
          a00(iel,2,2) =  k1(iel)+fk(iel)+0.5*deltat*fl(iel)
          a00(iel,3,3) =  k1(iel)+fk(iel)+0.5*deltat*fl(iel)
c
          a11(iel,1,1) =  (2.0d0 * xmut(iel) + xlambdat(iel))
c
     &                    * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,2)))))
     &                    * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,1))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,1)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,1))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,1)
c
          a22(iel,1,1) =  xmut(iel)
c
     &                    * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,3)))))
     &                    * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,1)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,1))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,1)
c
          a12(iel,1,2) =  xlambdat(iel)
c
     &                    * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,2)))))
     &                    * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,1))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,1)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,1))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,1)
```

```
c
      a21(iel,1,2) =  xmut(iel)
c
     &                   * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                   * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                   * (1.0d0 + fevn(iel,3)))))
     &                   * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
c
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                   * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                   * (1.0d0 + fevn(iel,1)))))**(-1)
c
     &                   * (1.0d0 + fevn(iel,2))
     &                      + 0.5*deltat*cs(iel)*fprp(iel,2)
c
      a33(iel,1,1) =  xmut(iel)
c
     &                   * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                   * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                   * (1.0d0 + fevn(iel,1)))))
     &                   * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,3))
c
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                   * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                   * (1.0d0 + fevn(iel,1)))))**(-1)
c
     &                   * (1.0d0 + fevn(iel,1))
     &                      + 0.5*deltat*cs(iel)*fprp(iel,1)
c
      a13(iel,1,3) =  xlambdat(iel)
c
     &                   * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                   * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                   * (1.0d0 + fevn(iel,2)))))
     &                   * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,1))
c
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                   * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                   * (1.0d0 + fevn(iel,1)))))**(-1)
c
     &                   * (1.0d0 + fevn(iel,1))
     &                      + 0.5*deltat*cs(iel)*fprp(iel,1)
c
      a31(iel,1,3) =  xmut(iel)
c
     &                   * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                   * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                   * (1.0d0 + fevn(iel,1)))))
     &                   * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,3))
c
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                   * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                   * (1.0d0 + fevn(iel,1)))))**(-1)
c
     &                   * (1.0d0 + fevn(iel,3))
     &                      + 0.5*deltat*cs(iel)*fprp(iel,3)
c
c
      a11(iel,2,2) =  xmut(iel)
c
```

```
     &                    * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,2)))))
     &                    * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,1))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,2)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,2))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,2)
c
          a22(iel,2,2) =  (2.0d0 * xmut(iel) + xlambdat(iel))
c
     &                    * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,3)))))
     &                    * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,2))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,2)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,2))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,2)
c
          a12(iel,2,1) =  xmut(iel)
c
     &                    * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,2)))))
     &                    * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,1))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,2)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,1))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,1)
c
          a21(iel,2,1) =  xlambdat(iel)
c
     &                    * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,3)))))
     &                    * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,2))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,2)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,2))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,2)
c
          a33(iel,2,2) =  xmut(iel)
c
     &                    * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,1)))))
```

```
     &                    * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,2)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,2))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,2)
c
          a23(iel,2,3) =  xlambdat(iel)
c
     &                    * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,3)))))
     &                    * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,2))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,2)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,2))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,2)
c
          a32(iel,2,3) =  xmut(iel)
c
     &                    * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,1)))))
     &                    * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,2)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,3))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,3)

c
          a13(iel,3,1) =  xmut(iel)
c
     &                    * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,2)))))
     &                    * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,3)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,1))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,1)
c
          a31(iel,3,1) =  xlambdat(iel)
c
     &                    * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,1)))))
     &                    * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,3))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,3)
```

```
     &                    * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,3)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,3))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,3)
c
          a23(iel,3,2) =  xmut(iel)
c
     &                    * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,3)))))
     &                    * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,2))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,3)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,2))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,2)
c
          a32(iel,3,2) =  xlambdat(iel)
c
     &                    * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,1)))))
     &                    * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,3))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,3)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,3))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,3)
c
          a11(iel,3,3) =  xmut(iel)
c
     &                    * ((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,2)))))
     &                    * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,3)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,3))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,3)
c
          a22(iel,3,3) =  xmut(iel)
c
     &                    * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,3)))))
     &                    * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,2))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,3)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,3))
```

```
     &                      + 0.5*deltat*cs(iel)*fprp(iel,3)
c
          a33(iel,3,3) =  (2.0d0 * xmut(iel) + xlambdat(iel))
c
     &                    * ((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                    * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                    * (1.0d0 + fevn(iel,1)))))
     &                    * ((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,3))
c
     &                    + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                    * (1.0d0 + fevn(iel,3)))))**(-1)
c
     &                    * (1.0d0 + fevn(iel,3))
     &                       + 0.5*deltat*cs(iel)*fprp(iel,3)

      else
c
          a00(iel,1,1) =  xrhot(iel)*tc1(iel)
          a00(iel,2,2) =  xrhot(iel)*tc1(iel)
          a00(iel,3,3) =  xrhot(iel)*tc1(iel)
c
          a11(iel,1,1) =  2.0d0 * xmut(iel) + xlambdat(iel)
          a22(iel,1,1) =  xmut(iel)
          a12(iel,1,2) =  xlambdat(iel)
          a21(iel,1,2) =  xmut(iel)
          a33(iel,1,1) =  xmut(iel)
          a13(iel,1,3) =  xlambdat(iel)
          a31(iel,1,3) =  xmut(iel)
c
          a11(iel,2,2) =  xmut(iel)
          a22(iel,2,2) =  2.0d0 * xmut(iel) + xlambdat(iel)
          a12(iel,2,1) =  xmut(iel)
          a21(iel,2,1) =  xlambdat(iel)
          a33(iel,2,2) =  xmut(iel)
          a23(iel,2,3) =  xlambdat(iel)
          a32(iel,2,3) =  xmut(iel)
c
          a13(iel,3,1) =  xmut(iel)
          a31(iel,3,1) =  xlambdat(iel)
          a23(iel,3,2) =  xmut(iel)
          a32(iel,3,2) =  xlambdat(iel)
          a11(iel,3,3) =  xmut(iel)
          a22(iel,3,3) =  xmut(iel)
          a33(iel,3,3) =  2.0d0 * xmut(iel) + xlambdat(iel)
c
      endif
c
8101   continue
c
      endif
c
c
c------ load up the rhs coefficients
c
      if(iresid .eq. GETRHSORRESIDUAL .or. passnum .eq. GETLHS) then
c
        if (pml_on .eq. 1) then
c
        do 9101 iel = ielstrt,ielend
          do 9111 icomp = 1,nca
c
             f(iel,icomp) = (k1(iel) * u(iel,icomp,soltn)
```

```
     &                     + k2(iel) * u(iel,icomp,veltn)
     &                     + k3(iel) * u(iel,icomp,acctn)
     &                     - fl(iel)*deltat*sum_u(iel,icomp,soltn))
c
9111     continue
c
             fx(iel,1) = -(((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                     * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                     * (1.0d0 + fevn(iel,2)))))
     &                     *Tcur(iel,1,1)
     &                   + (1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
     &                     *deltat*sum_T(iel,1,1)
     &                   + ((cs(iel)**2)*fprp(iel,2)*fprp(iel,3))
     &                      *(deltat**2)*sum2_T(iel,1,1))
c
             fx(iel,2) = -(((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                     * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                     * (1.0d0 + fevn(iel,3)))))
     &                     *Tcur(iel,1,2)
     &                   + (1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
     &                     *deltat*sum_T(iel,1,2)
     &                   + ((cs(iel)**3)*fprp(iel,2)*fprp(iel,1))
     &                      *(deltat**2)*sum2_T(iel,1,2))
c
             fx(iel,3) = -(((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                     * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                     * (1.0d0 + fevn(iel,1)))))
     &                     *Tcur(iel,1,3)
     &                   + (1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
     &                     *deltat*sum_T(iel,1,3)
     &                   + ((cs(iel)**1)*fprp(iel,2)*fprp(iel,2))
     &                      *(deltat**2)*sum2_T(iel,1,3))
c
c             print *,'fx',fx
c
             fy(iel,1) = -(((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                     * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                     * (1.0d0 + fevn(iel,2)))))
     &                     *Tcur(iel,2,1)
     &                   + (1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
     &                     *deltat*sum_T(iel,2,1)
     &                   + ((cs(iel)**2)*fprp(iel,2)*fprp(iel,3))
     &                      *(deltat**2)*sum2_T(iel,2,1))
c
             fy(iel,2) = -(((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                     * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                     * (1.0d0 + fevn(iel,3)))))
     &                     *Tcur(iel,2,2)
     &                   + (1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
     &                     *deltat*sum_T(iel,2,2)
     &                   + ((cs(iel)**3)*fprp(iel,2)*fprp(iel,1))
     &                      *(deltat**2)*sum2_T(iel,2,2))
c
             fy(iel,3) = -(((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
     &                   + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                     * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                     * (1.0d0 + fevn(iel,1)))))
     &                     *Tcur(iel,2,3)
     &                   + (1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
```

```
     &                   *deltat*sum_T(iel,2,3)
     &                 + ((cs(iel)**1)*fprp(iel,2)*fprp(iel,2))
     &                    *(deltat**2)*sum2_T(iel,2,3))
c
c             print *,'fy',fy
c
              fz(iel,1) = -(((1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
     &                 + 0.5*deltat*(cs(iel)*(fprp(iel,2)
     &                   * (1.0d0 + fevn(iel,3)) + fprp(iel,3)
     &                   * (1.0d0 + fevn(iel,2)))))
     &                   *Tcur(iel,3,1)
     &                 + (1.0d0 + fevn(iel,2))*(1.0d0 + fevn(iel,3))
     &                   *deltat*sum_T(iel,3,1)
     &                 + ((cs(iel)**2)*fprp(iel,2)*fprp(iel,3))
     &                    *(deltat**2)*sum2_T(iel,3,1))
c
              fz(iel,2) = -(((1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
     &                 + 0.5*deltat*(cs(iel)*(fprp(iel,3)
     &                   * (1.0d0 + fevn(iel,1)) + fprp(iel,1)
     &                   * (1.0d0 + fevn(iel,3)))))
     &                   *Tcur(iel,3,2)
     &                 + (1.0d0 + fevn(iel,3))*(1.0d0 + fevn(iel,1))
     &                   *deltat*sum_T(iel,3,2)
     &                 + ((cs(iel)**3)*fprp(iel,2)*fprp(iel,1))
     &                    *(deltat**2)*sum2_T(iel,3,2))
c
              fz(iel,3) = -(((1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
     &                 + 0.5*deltat*(cs(iel)*(fprp(iel,1)
     &                   * (1.0d0 + fevn(iel,2)) + fprp(iel,2)
     &                   * (1.0d0 + fevn(iel,1)))))
     &                   *Tcur(iel,3,3)
     &                 + (1.0d0 + fevn(iel,1))*(1.0d0 + fevn(iel,2))
     &                   *deltat*sum_T(iel,3,3)
     &                 + ((cs(iel)**1)*fprp(iel,2)*fprp(iel,2))
     &                    *(deltat**2)*sum2_T(iel,3,3))
c
c             print *,'fz',fz
c
9101   continue
c
         else
c
        do 9151 icomp = 1,nca
          do 9161 iel = ielstrt,ielend
c
              f(iel,icomp) = xrhot(iel) * (
     &          u(iel,icomp,soltn) / (beta * deltat**2) +
     &          u(iel,icomp,veltn) / (beta * deltat)    +
     &          u(iel,icomp,acctn) * (1.0/(2.0*beta) - 1.0) )
c
             fx(iel,icomp) = 0.0d0
             fy(iel,icomp) = 0.0d0
             fz(iel,icomp) = 0.0d0
c
9161     continue
9151   continue
c
       endif
      endif
c
c
      return
      end
c
```

```
c========================================================= strain ======
c
      subroutine strain(uxyz, numel, ncomp, nca, nsol, deltat, ielstrt,
     .              ielend, fprp, fevn, cs,
     .              Ecur, sum_e, sum2_e, sum_uij, etn, etnp1, curstep)
c
c*********************************************************************
c*                                                                   *
c*  Routine:  strain                                                 *
c*  Purpose:  computes the strain solution, velocity and             *
c*            acceleration vectors at time T(n+1) based on the       *
c*            solution, velocity, and acceleration at time T(n).     *
c*                                                                   *
c*     Parameters:                                                   *
c*       I   uxyz      -                                           *
c*       I   Ecur      - current strain                            *
c*       I   etn       - strain at time Tn                      *
c*       I   etnp1     - strain at time Tn+1                       *
c*                                                                   *
c*                                                                   *
c*********************************************************************
c
#include "PH-std.blk"
c
c
c
  INTTYPE numel, ncomp, nca, nsol, iel, ielstrt, ielend
  INTTYPE icomp, jcomp, k, soltn, veltn, acctn, curstep
c
      REALTYPE deltat
      REALTYPE uxyz(numel,ncomp,nsol,3),
     &         Ecur(MAXELEMBATCH,ncomp,ncomp),
     &         etn(MAXELEMBATCH,ncomp,ncomp),
     &         etnp1(MAXELEMBATCH,ncomp,ncomp),
     &         sum_e(MAXELEMBATCH,ncomp,ncomp),
     &         sum2_e(MAXELEMBATCH,ncomp,ncomp),
     &         sum_uij(MAXELEMBATCH,ncomp,ncomp),
     &         fprp(MAXELEMBATCH,ncomp),
     &         fevn(MAXELEMBATCH,ncomp),
     &         cs(MAXELEMBATCH)
c
c      print *,'inside strain subroutine....'
c
c-------------------------------------------------------------------
c
      call PH_UG_GET_SLOT_(SOLUTION_TN, soltn)
      call PH_UG_GET_SLOT_(VELOCITY_TN, veltn)
      call PH_UG_GET_SLOT_(ACCELERATION_TN, acctn)
c
c
        etn = Ecur
  sum_e = sum_e + etn
c
        sum2_e = (curstep + curstep**2)*etn
c
      do 3001 iel = ielstrt,ielend
c
c------ building sum vector
c
       sum_uij(iel,1,1) = (1.0d0*sum_uij(iel,1,1) +
     &   ( fprp(iel,1)*uxyz(iel,1,soltn,1)
     &   + fprp(iel,1)*uxyz(iel,1,soltn,1)) )
c
       sum_uij(iel,1,2) = (1.0d0*sum_uij(iel,1,2) +
```

```
     &   ( fprp(iel,1)*uxyz(iel,1,soltn,2)
     &   + fprp(iel,2)*uxyz(iel,2,soltn,1)) )
c
       sum_uij(iel,1,3) = (1.0d0*sum_uij(iel,1,3) +
     &   ( fprp(iel,1)*uxyz(iel,1,soltn,3)
     &   + fprp(iel,3)*uxyz(iel,3,soltn,1)) )
c
       sum_uij(iel,2,1) = (1.0d0*sum_uij(iel,2,1) +
     &   ( fprp(iel,2)*uxyz(iel,2,soltn,1)
     &   + fprp(iel,1)*uxyz(iel,1,soltn,2)) )
c
       sum_uij(iel,2,2) = (1.0d0*sum_uij(iel,2,2) +
     &   ( fprp(iel,2)*uxyz(iel,2,soltn,1)
     &   + fprp(iel,2)*uxyz(iel,2,soltn,2)) )
c
       sum_uij(iel,2,3) = (1.0d0*sum_uij(iel,2,3) +
     &   ( fprp(iel,2)*uxyz(iel,2,soltn,3)
     &   + fprp(iel,3)*uxyz(iel,3,soltn,2)) )
c
       sum_uij(iel,3,1) = (1.0d0*sum_uij(iel,3,1) +
     &   ( fprp(iel,3)*uxyz(iel,3,soltn,1)
     &   + fprp(iel,1)*uxyz(iel,1,soltn,3)) )
c
       sum_uij(iel,3,2) = (1.0d0*sum_uij(iel,3,2) +
     &   ( fprp(iel,3)*uxyz(iel,3,soltn,2)
     &   + fprp(iel,2)*uxyz(iel,2,soltn,3)) )
c
       sum_uij(iel,3,3) = (1.0d0*sum_uij(iel,3,3) +
     &   ( fprp(iel,3)*uxyz(iel,3,soltn,3)
     &   + fprp(iel,3)*uxyz(iel,3,soltn,3)) )
c
c------ build strain vector for t(n+1)
c
      do 9201 icomp = 1,nca
        do 9211 jcomp = 1,nca
c
      etnp1(iel,icomp,jcomp) = 0.5d0
     &    *((1.0d0 + fevn(iel,icomp))*(1.0d0 + fevn(iel,jcomp))
     &  + 0.5*deltat*(cs(iel)*(fprp(iel,icomp)*(1.0d0 + fevn(iel,jcomp))
     &              + fprp(iel,jcomp)*(1.0d0 + fevn(iel,icomp)))))**(-1)
     &    * (1.0d0*cs(iel)*deltat*sum_uij(iel,icomp,jcomp)
     &  - 2.0d0*deltat*(cs(iel)*(fprp(iel,icomp)*(1.0d0 + fevn(iel,jcomp))
     &              + fprp(iel,jcomp)*(1.0d0 + fevn(iel,icomp))))
     &    * sum_e(iel,icomp,jcomp)
     &  - 2.0d0*deltat**2*((cs(iel)**2)*fprp(iel,icomp)*fprp(iel,jcomp))
     &    * sum2_e(iel,icomp,jcomp) )
c
 9211   continue
 9201 continue
c
3001   continue
c
       Ecur = etnp1
c    print *, Ecur
c
c
      return
      end
```

APPENDIX. F (PROPHLEX C++ MODULE)

```
/* --- application.c ---- Fri Jul 26 14:29:13 CDT 1996 ----
 *  Copyright: Computational Mechanics, Co., Inc. 1992-1996
 *  MAJ Anthony N. Johnson, Naval Postgraduate School
 *  Monterey, California 93940
==========================================================
**********************************************************/
/* ###################################################################### */
/* File: application.c
 * ======================== */

#include "PH-std.h"
#include <string.h>

/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* Function Prototypes
 * ------------------- */

#include "PH-util.h"
#include "PH-register.h"
#include "PH-parameters.h"
#include "PH-adapt.h"

#define   APPLICATION   15       /* the application number */

#include "PH-Fmacros.h"
#include "PH-post.h"
#include "PH-gui.h"
#include "PH-tcl.h"
#include "PH-io.h"
#include "PH-results-save.h"

#include  "_release_name.h"
#include  "application-parameters.h"
#include  "application-macros.h"
#include  "application-main.h"
#include  "application-post.h"
#include  "application-FtoC.h"
#include  "applicationFile-Register.h"

/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* Global variable
 * --------------- */

extern Tcl_Interp * tcl;


/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* Internal Function Prototypes
 * ---------------------------- */

static void application_tcl(void);
static int  cmdApplication(ClientData clientData, Tcl_Interp * interp,
   int argc, char *argv[]);


/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* Static Variables
 * ---------------- */

/* define postprocessor components and
```

```
 *  type of data needed for evaluation
 */

static struct PostComp postApplication[] =
{
  {XCOMP, "X", NEED_SOLONLY},
  {YCOMP, "Y", NEED_SOLONLY},
  {ZCOMP, "Z", NEED_SOLONLY},

  {PRIMARY1,   "X Displacements", NEED_SOLONLY},
  {PRIMARY2,   "Y Displacements", NEED_SOLONLY},
  {PRIMARY3,   "Z Displacements", NEED_SOLONLY},

  {SECONDARY1,   "Secondary 1", NEED_DERIV},
  {SECONDARY2,   "Secondary 2", NEED_DERIV},
  {SECONDARY3,   "Secondary 3", NEED_DERIV},
  {NEWCOMP   ,   "New Comp"   , NEED_DERIV},
  {NEWCOMPWAVE , "Wave Comp"  , NEED_DERIV},

  {ESTIMATE, "Error Indicator", ELEMENT_COMP},

  /* the following is always last */
  {0, '\0', 0}
};


/* ################################################################## */
/* declaration of structures for HM output file formats */

static int Num_Results_Fmts = 1 ;

static ResultsFileFmt Sav_Results_Fmts[]=
{

{"res", "HM", "*.res", 1,
 {{"",0}},
   FM_BINARY,
   PH_WriteHM_Results,
   PH_WriteHM_Mesh,
   PH_WriteHM_All,
   PH_AppendHM_Step,
   PH_FinishHM_Steps,

  9,

 {
 {  XCOMP       ,  RF_SCALAR   ,          1, "X"},
 {  YCOMP       ,  RF_SCALAR   ,          2, "Y"},
 {  ZCOMP       ,  RF_SCALAR   ,          3, "Z"},

 {  PRIMARY1    ,  RF_VECTOR   ,          4, "Displacements"},
 {  PRIMARY2    ,  RF_VECTOR   ,          4, "Displacements"},
 {  PRIMARY3    ,  RF_VECTOR   ,          4, "Displacements"},

 {  SECONDARY1  ,   RF_SCALAR  ,          5, "Secondary 1"},
 {  SECONDARY2  ,   RF_SCALAR  ,          6, "Secondary 2"},
 {  SECONDARY3  ,   RF_SCALAR  ,          7, "Secondary 3"},
 {  NEWCOMP     ,   RF_SCALAR  ,          8, "New Comp"   },
 {  NEWCOMPWAVE ,   RF_SCALAR  ,          9, "Wave Comp"   }

 /* Vector components are also possible

 {  SOL1       ,   RF_VECTOR   ,          4, "Displacements"},
 {  SOL2       ,   RF_VECTOR   ,          4, "Displacements"},
```

```
 {  SOL3       ,   RF_VECTOR   ,           4, "Displacements"},

 */

 }
}
};

/* ###################################################################### */


/* Define the solution algorithm name and potential linear solvers
 * to call.   Note, the alorithm integer reference number for
 * ALG_TEMPLATE is stored under KP_ALGORITHM in the parameter set
 * KERNEL_PARAMS.  This parameter may be used to branch between
 * algorithms in a given application e.g. (linear, nonlinear, transient ...)
 */

static Alg_Data_Type algApplication[] =
{
  {ALG_TEMPLATE, "Your algorithm",
   {SPARSE_OPT, SPARSE_OPT, SPARSE_OPT, SPARSE_OPT, SPARSE_OPT}},
  {0, '\0'}
};

/* define the sequence of buttons and names
 *  to appear on the viewport window
 */

static struct PostTopbar post_menu[] = {
   {CONTROL_FORM,         "Control"},
   {COLOR_EDIT_FORM,      "COLOR2",    "COLOR2"},
   {SET_SELECT_FORM,      "SETS",      "SETS"},
   {VIEW_FORM,            "EYE2"},
   {COMP_SELECTOR_FORM,   "LoadResults"},
   {SLICE_FORM,           "Slice"},
   {ISOSURF_FORM,         "Iso"},
   {XY_FORM,              "XYplot"},
   {PROBE_FORM,           "Probe"},
   {LOADS_PLOT_FORM,      "BCs"},
   {HARDCOPY_FORM,        "Hardcopy"},
   {DEFRM_FORM,           "Deform"},
   {VELVEC_FORM,          "VelVec"},
   {TRACES_FORM,          "Traces"},
   {RESET_ACTION,         "ResetView"},
   {0, NULL},
};

/*########################################################################## */
/*             E X P O R T A B L E     F U N C T I O N S                  */
/* ########################################################################## */
int main ( int argc, char ** argv )
{
  return PH_PhlexMain ( argc, argv );
}
/* ###################################################################### */
void PH_InitializeApplicationDB(void)
{

/*
 * This routine is the main registration and initialization routine
 * called by default from the kernel
 */
```

```
/* set the icon name to appear on the main form and
 * the release name
 */

  PH_Gui_SetMainForm ( "ProPHLEX", NULL );

  PH_Gui_SetReleaseName(_RELEASE_NAME);

  applicationFile_RegisterObjDB ();  /* attach user objects to database */

/* Register the following;
 *   application name, initialization routine
 *   postprocessor components to evaluate
 *   HM interface options
 *   linear equation solver to link
 *   tcl initialization routine for special application parameters
 */

  PH_RegisterProblem(APPLICATION, "Template",
   SCOPE, (void *) &algApplication);

  PH_RegisterPostcomp(APPLICATION, (void *) &postApplication, 0, PRIMARY1,
    NULL , NULL);

  PH_RegisterResultsFmts (APPLICATION, Num_Results_Fmts, Sav_Results_Fmts);

  PH_RegisterSolver( SPARSE_OPT, "Sparse" , PH_INIT_SPARSE_ ); /* sparse solver */

  PH_RegisterTclInitialization((FUNPTR) application_tcl);

/* Register the following:
 *    the first slot to use for velocity vectors in post
 *         note for deformed plots the first three slots are used by default
 */


  PH_Gui_SetPostOption (POST_VECTOR_COMP_ID, PRIMARY1);


  fprintf(PH_dbgout, "\n\n");
  fprintf(PH_dbgout, ">>>> ================================== <<<<\n");
  fprintf(PH_dbgout, ">>>>        WELCOME TO SAFE-T           <<<<\n");
  fprintf(PH_dbgout, ">>>>  hp-ADAPTIVE FINITE ELEMENT ANALYSIS <<<<\n");
  fprintf(PH_dbgout, ">>>>    TOOL using the PHLEX kernel       <<<<\n");
  fprintf(PH_dbgout, ">>>> ================================== <<<<\n");
  fprintf(PH_dbgout, "\n\n");

}

/* ###################################################################### */
void PH_InitializeApplicationParams( void )
{
/* this routine initializes and set default values for;
 *    the aplication parameters
 *    the material data base parameters
 *    the entries to appear on the veiwport window
 */

  SetDefaultApplicationParams();
  SetDefaultMaterialDB();
  SetDefaultBoundaryConditionsDB();
  PH_Gui_SetPostMainMenu ( post_menu );

  /* turn on initial mesh conditioning */
```

```
  PH_Gui_ConditionMesh("on",(double)0);

}

/* ###################################################################### */
static void application_tcl(void)
{

/*
 * initialization of application specific tcl options
 * in particular, the function cmdApplication below is registered
 * as a call back function from the kernel
 */

    Tcl_CreateCommand(tcl, "command", cmdApplication, (ClientData) NULL,
      (Tcl_CmdDeleteProc *) NULL);
}

/* ###################################################################### */
static int cmdApplication( ClientData clientData,
                    Tcl_Interp *interp,
                    int argc,
                    char *argv[] )
{

/*
 * application specific tcl options are defined in this routine
 */

    char ActionType[MAX_PARAM_NAME_LEN];
    int action;

    if (argc != 3) {
Tcl_AppendResult(interp, "wrong # args: should be \"", argv[0],
 " action_type well_number\"", (char *) NULL);
return TCL_ERROR;
    }

    strcpy(ActionType, argv[1]);

    if (strcmp(ActionType, "token") == STRING_MATCH) {
action = YES;
    } else {
Tcl_AppendResult(interp, "unable to identify action type",
 (char *) NULL);
return TCL_ERROR;
    }

    return TCL_OK;

}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. G (DISCRETE CONVOLUTION USING MATLAB)

```
clear M;clear a;clear f;clear h;clear p; close all
a=0.005;
j=0;
etime=2;
step=0.01;
time=0:step:etime;
length(time);
for s = 1:length(time)
    j = j+1;
%       p(j) = -0.055;
%     p(j) = -2*sourceGaus(time(s));
    p(j) = -2.5*derGaus(time(s));
end

r=1;
length(p);
M=zeros(length(p));
length(p);
[m,n]=size(M);
for i = 1:m
    for j = 1:n
%         M(i,j)=Heavi(i-j)-Heavi(i-j-1);
        h(j) = wtime(time(j),r);
        M(i,j)=wtime(time(j)-time(i),r)-wtime(time(j)-time(i)-step,r);
    end
end

f=p*M;

subplot(3,1,1);plot(time,f)
grid
title('Surface Response 1m away')
ylabel('Displacement')

subplot(3,1,2);plot(time,p,'r');
grid
title('Surface Source')
```

```
ylabel('Displacement')

subplot(3,1,3);plot(time,h,'g');
grid
title('Pekeris 1955')
xlabel('time (seconds)')
ylabel('Displacement')


figure(2)
plot(time,f)
grid
title('Surface Response 1m away')
ylabel('Displacement')
```

LIST OF REFERENCES

[1] H. Lamb. On the propagation of tremors over the surface of an elastic solid. *Phil. Trans. Royal Soc*, 203:1–42, 1904.

[2] C. L. Pekeris. The seismic surface pulse. *Proc. Natl. Acad. Sci*, 41:469–475, 1955.

[3] W. W. Garvin. Exact transient solution of the burried line source problem. *Proc. Royal Soc.*, 234:528–541, 1956.

[4] K. F. Graff. *Wave Motion in Elastic Solids*. Dover Publishing Inc, New York, 1975.

[5] J. D. Achenbach. *Wave Propagation in Elastic Solids*. Elsevier Science Publisher B.V., North-Holland, 1975.

[6] H. M. Mooney. Some numerical solutions for lamb's problem. *Bulletin of the Seismological Society of America*, 64:473–491, 1974.

[7] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method*. Butterworth-Heinemann, Woburn, MA, 2000.

[8] Y. Q. Zeng and Q. H. Liu. Acoustic landmine detection: a 3d poroelastic model. *Proc. SPIE, Orlando FL*, 2001.

[9] S. E. Rumph. Development of four-element end-fire array as seismo-acoustic sonar source. *Naval Postgraduate School*, Master's Thesis:1–55, 2003.

[10] G. L. Fenves and A. K. Chopra. Earthquake analysis and response of concrete gravity dams. *University of California, Berkley, CA, USA*, 1984.

[11] U. Basu and A. K. Chopra. Perfectly matched layers for time-harmonic elastodynamics of unbounded domains: Theory and finite-element implementation. *Computer Methods in Applied Mechanics and Engineering*, 192:1337–1375, 2003.

[12] R. Clayton and B. Engquist. Absorbing boundary conditions for acoustic and elastic wave equations. *Bulletin Seis. Soc. Amer.*, 67:1529–1540, 1977.

[13] D. Bernal and A. Youssef. A hybrid time frequency domain formulation for non-linear soil-structure interaction. *Eathquake Engineering and Structural Dynamics*, 27:673–685, 1998.

[14] U. Basu and A. K. Chopra. Perfectly matched layers for transient elastodynamics of unbounded domains. *International Journal for Numerical Methods in Engineering*, 59:12–14, 2004.

[15] J. P. Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, 114:185–200, 1994.

[16] W. H. Weedon W. C. Chew. A 3d perfectly matched medium from modified maxwell's equations with stretched coordinates. *Microwave and Optical Technology Letters*, 7:599–604, 1994.

[17] T. V. Yioultsis T. I. Kosmanis and T. D. Tsiboukis. Perfectly matched anisotropic layer for the numerical analysis of unbounded eddy-current problems. *IEEE Trsnsctions On Magnetics*, 35:4452–4458, 1999.

[18] S. A. Cummer. A simple, nearly perfectly matched layer for general electromagnetic media. *IEEE Microwave and Wireless Components Letters*, 13:128–130, 2003.

[19] B. Neta I. M. Navon and M. Y. Hussaini. A perfectly matched layer approach to the linearized shallow water equations models. *American Meteorological Society*, 132:1369–1378, 2004.

[20] W. C. Elmore and M. A. Heald. *Physics of Waves*. Dover Publications, Inc., New York, 1985.

[21] G. T. Mace and G. E. Mase. *Continuum Mechanics for Engineers*. CRC Press, New York, 1999.

[22] Y. C. Fung. *Foundations of Solid Mechanics*. Prentice Hall, Inc., New Jersey, 1965.

[23] G. Genta. *Vibration of Structures and Machines*. Maple-Vail Book Manufactoring Group, York, PA, 1993.

[24] L. Brekhovskikh and V. Goncharov. *Mechanics of Continua and Wave Dynamics*. Springer-Verlag, New York, 1994.

[25] L. Debnath. *Integral Transforms and Their Applications*. CRC Press, New York, 1999.

[26] R. Haberman. *Applied Partial Differential Equations with Fourier Series and Boundary Value Problems*. Pearson Prentice Hall, New Jersey, 2004.

[27] A. C. Eringen and E. S. Suhubi. *Elastodynamics, Volume 2*. Academic Press, New York, 1975.

[28] D. Vamvatsikos H. G. Georgiadis and I. Vardoulakis. Numerical implementation of the integral-transform solution to lamb's point-load problem. *Computational Mechanics*, 24:90–99, 1999.

[29] G. F. Miller and H. Pursey. Partition of energy between elastic waves. *Proceedings Royal Soc*, 223:55–69, 1955.

[30] R. D. Wood. Screeinig of surface waves in solids. *Journal of Soil Mechanics and Foundations Division, ASCE*, 4:951–979, 1968.

[31] R. N. Bracewell. *McGraw Hill Electrical And Electronic Engineering Series*. McGraw Hill, New York, 1978.

[32] E. O. Brigham. *The Fast Fourier Transform*. Printice Hall, New Jersey, 1974.

[33] D. S. Burnett. *Finite Element Analysis. From Concepts to Applications*. Addison-Wesley, Reading, Mass., 1987.

[34] W. A. Strauss. *Partial Differential Equations, An Introduction*. John Wiley and Sons, Inc., New York, 1992.

[35] R. D. Richtmyer and K. W. Morton. *Differencr Methods for Initial Value Problems, 2nd ed*. Wiley, New York, 1967.

[36] E. Isaacson and H. B. Keller. *Analysis of Numerical Methods*. John Wiley and Sons, Inc., New York, 1966.

[37] G. Dahlquist. Convergence and stability in the numerical integration of ordinary differential equations. *Math. Scand.*, 4:33–53, 1956.

[38] J. D. Lambert. *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. John Wiley and Sons, New York, 1932.

[39] M. Ross. Modeling methods for silent boundaries in infinite media. *Fluid-Structure Interaction, Aerospace Engineering Sciences-University of Colorado at Boulder*, Report ASEN 5519-006, 2004.

[40] F. Q. Hu. Development of pml absorbing boundary condition for computational aeroacoustics: a progress review. *Department of Mathematics and Statistics, Old Dominion University, Norfolk, VA 23529*, Technical Report.

[41] G. Festa and S. Nielsen. Pml absorbing boundaries. *Bulletin of the Seismological Society of America*, 93:891–903, 2003.

[42] D. Komatitsch and J. Tromp. A perfectly matched layer absorbing boundary condition for the second-order seismic wave equation. *Goephys. J. Int.*, 154:146–153, 2003.

[43] L. Zhao and AC Cangellaris. A general approach for the development of unsplit-field time-domain implementation of perfectly matched layers for fdtd grid truncation. *IEEE Microwave and Guided Wave Letters*, 6, 1996.

[44] J. M. Bass S. K. Sharma T. A. Westerman B. B. Yavari T. J. Liszka, W. W. Tworzydlo. Prophlex - an hp-adaptive finite element kernel for solving coupled systems of partial differential equations in computational mechanics. *Comput. Methods Appl. Mech. Eng.*, 150:251–271, 1997.

[45] C. F. Gerald and P. O. Wheatly. *Applied Numerical Analysis*. Pearson Addison Wesley, New York, 2004.

[46] http://www.lngs.infn.it. Underground seismic array experiments. *National Institute of Nuclear Physics*, 2005.

[47] D. J. MacLean. Mobile source development for seismic-sonar based landmine detection. *Naval Postgraduate School*, Master's Thesis, 2003.

[48] W. F. Stewart. Buried object detection using surface waves. *Naval Postgraduate School*, Master's Thesis, 1995.

[49] F. E. Gaghan. Discrete mode source development and testing for new seismo-acoustic sonar. *Naval Postgraduate School*, Master's Thesis, 1998.

[50] S. M. Fitzpatrick. Source development for a seismo-acoustic sonar. *Naval Postgraduate School*, Master's Thesis, 1998.

[51] P. Hall. Detection and target strength measurements of buried objects using a seismic sonar. *Naval Postgraduate School*, Master's Thesis, 2002.

[52] K. E. Sheetz. Advancements in buried mine detection using seismic sonar. *Naval Postgraduate School*, Master's Thesis, 2002.

[53] S. C. McClelland. A rolling line source for a seismic sonar. *Naval Postgraduate School*, Master's Thesis, 2002.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Dr. Clyde Scandrett
   Naval Postgraduate School
   Monterey, CA

4. Dr. Steve Baker
   Naval Postgraduate School
   Monterey, CA

5. Dr. Beny Neta
   Naval Postgraduate School
   Monterey, CA

6. Dr. Dave Olwell
   Naval Postgraduate School
   Monterey, CA

7. Dr. Carlos F. Borges
   Naval Postgraduate School
   Monterey, CA